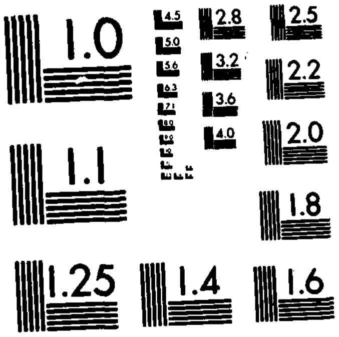
AD-A124 647 ANALYSIS OF INFORMATION REQUIREMENTS AND DESIGN OF THE 1/4
CONSOLIDATED AFIT D..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..
UNCLASSIFIED J S RICKS ET AL. DEC 82 AFIT/GCS/EE/82D-11 F/G 5/2 NL


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
1.1
1.25
1.4
1.6

83 02 022 147

0

Analysis of Information Requirements and Design of the Consolidated AFIT Database and Information System (CADIS) with an AFIT/CI Implementation Design

THESIS

Jeffrey S. Ricks
Capt USAF
Robert S. Colburn
2Lt USAF

AFIT/GCS/EE/82D-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

83 02 022 147

Best available copy
per the originator

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
AFIT/GCS/EE/82D-11

2. GOVT ACCESSION NO.
AD-A124647

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
Analysis of Information Requirements and Design of the Consolidated AFIT Database and Information System (CADIS) with an AFIT/CI Implementation

5. TYPE OF REPORT & PERIOD COVERED
AFIT MS Thesis

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Capt Jeffrey S. Ricks USAF
2Lt Robert S. Colburn USAF

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Air Force Institute of Technology (AFIT/EN)
Wright-Patterson AFB, Ohio 45433

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

12. REPORT DATE
December 1982

13. NUMBER OF PAGES
298

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved ~~for public release; IAW AFR 190-17~~

4 JAN 1983

Approved for public release: IAW AFR 190-17.
Lyn Wolaver
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Database	Evaluation	Software Engineering	Implementation
Relational	Analysis	Top-Down	Normalization
DBMS	Requirements	Structured	Management

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A Consolidated AFIT Database and Information System (CADIS), a centralized relational database in third normal form, was designed for use by the faculty and staff of the Air Force Institute of Technology (AFIT). Information requirements were gathered by means of an in-depth data/information analysis at the directorate and division level, and used as the basis of the design. Provided with the design is a highly detailed data

(Block 20)

dictionary, in four parts, and an estimation of the approximate size of the database itself.

In conjunction with the database design, all commercially available Database Management Systems (DBMS) were surveyed and evaluated for use by AFIT. A detailed evaluation procedure was designed, and documented, to reflect the information needs of AFIT and each DBMS was in-turn judged against it. The result was a list of commercially available DBMSs, ranked according to how they would fulfill the needs of AFIT.

As a result of certain facts gathered during the data/information analysis, several management and organizational recommendations were made to AFIT. The aim of these recommendations was to make the implementation and management of the database easier and more effective and to make the database more responsive to the users.

Additionally, a system of applications programs were designed for the AFIT Directorate of Civilian Institution Programs (AFIT/CI). This system will allow efficient and easy access to the data in the database by untrained or non-programmer type users. These programs were specifically designed to provide an extremely friendly user interface between CADIS and the AFIT/CI personnel. They have been conceptually and logically documented according to the standards of software engineering.

Analysis of Information Requirements and Design of the Consolidated AFIT Database and Information System (CADIS) with an AFIT/CI Implementation Design

THESIS

Jeffrey S. Ricks
Capt USAF
Robert S. Colburn
2Lt USAF

AFIT/GCS/EE/82D-11

AFIT/GCS/EE/82D-11

Analysis of Information Requirements and Design of the Consolidated AFIT Database and Information System (CADIS) with an AFIT/CI Implementation Design

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements of the Degree of
Master of Science

by
Jeffrey S. Ricks
Captain USAF
&
Robert S. Colburn
Lt. USAF
Graduate Computer Systems
December 1982

Approved for public release; distribution unlimited

PREFACE

The need for a centralized database and data management system has long been recognized by AFIT as essential for efficient operation in the face of an ever increasing student enrollment and the associated administrative load. Until the completion of this thesis, AFIT had been unable to complete the groundwork for a solution to their information management needs because of other problems associated with completely modernizing the AFIT data automation program.

To bring a project of this magnitude to completion and at the same time produce results which are accurate, useful and meaningful to AFIT required the combined efforts of many people. Grateful appreciation is extended to the AFIT Computer Configuration Board (CCB) and the heads of all the AFIT directorates and divisions who willingly and supportively participated in the data analysis interviews and to Ms. Jean Darjean from AFIT/ACD.

Heartfelt thanks is given to our thesis advisor Maj. Chuck Lillie and to committee member Dr. Henry Potoczny who provided tremendous moral support, innumerable impartial evaluations and recommendations and managed to keep us on track.

Special gratitude is extended to three people who, without their help, this project probably could not have been completed on time. Capt. Brian Van Ormann, AFIT/CI, spent many hours helping us understand the Civilian Institution Programs Directorate's requirements and problems. Capt. Ricardo Cuadros and Lt. Tim Mayberry, two fellow graduate students, lent their knowledge and manpower to the completion of the applications programs design (chapter IV).

Capt. Ricks would like to express his deepest appreciation to those fellow officers and civilians of the Air Force Data Services Center who provided him with the background and courage to undertake this type of project. I would especially like to thank LtCol. Marvin Lerfald who took a chance on a brand new 2Lt. and helped him mature both as an officer and as a professional data automator. I would also like to recognize Maj. Craig Goeller and the other members of the AFDSC/GKP PDS team for their friendship and professionalism as well as the knowledge we shared during the implementation of the FAS database and applications programs.

Lastly and certainly not least I would like to thank Lt. Colburn for his time and patience. These were difficult times and his cool head and academic demeanor kept me from perpetuating some very large mistakes and oversights. You have won my everlasting gratitude.

Lt. Colburn would like to express his deepest appreciation to his thesis partner Capt. Ricks without whose professional expertise, excellent background and endless hours of work, a thesis of this scope and magnitude could not have been accomplished. To put it in Navy terms Jeff, "Bravo Zulu", well done. I also owe a measureless debt to my wife, Jane, without whose support, encouragement and understanding on the home front I would have gone crazy. Finally, thanks to all the new friends I have made at AFIT, who were there with support and advice when I needed it.

ABSTRACT

A Consolidated AFIT Database and Information System (CADIS), a centralized relational database in third normal form, was designed for use by the faculty and staff of the Air Force Institute of Technology (AFIT). Information requirements were gathered by means of an in-depth data/information analysis at the directorate and division level, and used as the basis of the design. Provided with the design is a highly detailed data dictionary, in four parts, and an estimation of the approximate size of the database itself.

In conjunction with the database design, all commercially available Database Management Systems (DBMS) were surveyed and evaluated for use by AFIT. A detailed evaluation procedure was designed, and documented, to reflect the information needs of AFIT and each DBMS was in-turn judged against it. The result was a list of commercially available DBMSs, ranked according to how they would fulfill the needs of AFIT.

As a result of certain facts gathered during the data/information analysis, several management and organizational recommendations were made to AFIT. The aim of these recommendations was to make the implementation and management of the database easier and more effective and to make the database more responsive to the users.

Additionally, a system of applications programs were designed for the AFIT Directorate of Civilian Institution Programs (AFIT/CI). This system will allow efficient and easy access to the data in the database by untrained or non-programmer type users. These programs were speciffically designed to provide an extremely friendly user interface between

CADIS and the AFIT/CI personnel. They have been conceptually and logically designed and documented according to the standards of software engineering.

TABLE OF CONTENTS

I. INTRODUCTION

1. BACKGROUND

Various organizations within the Air Force Institute of Technology (AFIT) have expressed a requirement for automated management information systems to aid in the accomplishment of their mission requirements. Each year AFIT continues to expand and accept more students and offer more courses. Correspondingly there has been a rapid increase in the amount of paperwork and general administrative overhead associated with this growth. This is not unusual for an academic institution but the fact that this is also an Air Force organization presents some unique problems and situations.

Presently each school and department within AFIT maintains individual files and records on its students, faculty and courses in an arrangement unique to each. As a consequence, massive duplication of the type of data and in some cases actual data has occurred. Such duplication is wasteful both in terms of dollars and manpower. Utilization and maintenance of these systems has hindered and in many cases prevented effective interaction between departments and management activities.

Several directorates are beginning to realize, as has the rest of the Air Force and civilian industry, that a database management system, if properly implemented, can alleviate waste while increasing the effectiveness of everyday management. However, for each department to implement a database management system of their own would not solve anything, and to simply automate current problems would be even more wasteful. In order to avoid such a situation a single integrated database specifically designed for use within AFIT and fulfilling the requirements of all the departments must be implemented.

As an example of the extent of this problem, consider the AFIT Directorate of Civilian Institution Programs (AFIT/CI). They are responsible for the management of approximately 5000 AFIT students in civilian academic institutions, medical training and Education With Industry (EWI) programs. Currently all record keeping and retrieval of information is performed manually with file folders and hand scribed data sheets.

In spite of an ever increasing student load and a corresponding decline in resources, growing emphasis is being placed on conservation and "doing more with less". Requests for information of all types by upper management are increasing in number and importance and are now more critical to the daily operation of AFIT than ever before. These demands for more accurate and timely information have in general rendered present methods all but obsolete and are in fact quickly becoming a detriment. A simple request for information concerning the student body, either past or present, is complicated by the fact that several independent sources must be searched and the results compiled before any useful information can be produced. The expenditure in time and money is obviously tremendous and needlessly wasteful.

AFIT/CI, in conjunction with the remainder of AFIT, must automate their outdated method of accounting and record keeping in order to cope with present and future workloads and adequately respond to management requests for that information vital to the accomplishment of the AFIT mission.

1.1. PREVIOUS EFFORTS

In 1980 an attempt was made to test the feasibility for and possibly of designing a database which would be suitable for use throughout AFIT.(REF 4) This effort was aimed primarily at reducing data redundancy within AFIT by providing a central repository for information. Through the means of surveys and interviews, both resident schools were found to be maintaining a file system on the students assigned to them along with information about the faculty and courses offered.

This effort to properly design and implement an AFIT database was concerned primarily with the resident students, faculty and course data. It was to be performed in two phases; gathering of data requirements and design of the database.

Phase one consisted of interviews with the two AFIT departments, the School of Engineering (EN) and the School of Systems and Logistics (LS), in order to help them identify data requirements essential to the performance of their mission. Phase two consolidated these data elements and attempted to develop a combined database design which was supposed to fulfill their collective need.

Several departments within AFIT were not included in this effort, they were: Commandant (CC), Academic Affairs (CAE), Administration (DA), Public Affairs (PA), Academic Library (LD), Research and Professional Development (NR), School of Civil Engineering (DE) and the Civilian Institution Programs (CI). Of the omitted departments, one is of particular interest to this thesis, Civilian Institution Programs (CI). It was assumed that information required by each of these departments, although somewhat different, was similar enough that they could both utilize this "general" database if they modified their data requirements to fit it.

Since its completion in 1980, very little has been done in the area of actual implementation. Various portions of the proposed database have been set up for limited use by students in the engineering school, primarily to give students experience in interacting with an actual database. For reasons beyond the scope of this project, a full scale implementation of this database has yet to materialize, be loaded with meaningful data, and utilized throughout AFIT.

The shortcomings of the earlier effort are many and varied especially in light of current knowledge and goals. Those directorates which were not included in the original database design have started to move out on their own and seek automated support because of growing workloads. Consequently, they have been somewhat reluctant to participate in a database which does not accurately and completely fulfill their needs. In at least one instance, a directorate (AFIT/RR) has already acquired a registrar oriented file management system and is currently prepared to utilize it on a daily basis. It is not, however, a database management system and cannot conform to the overall AFIT goal of a single integrated database.

The first project was narrow in scope and did not include such vital areas as security and privacy of the database and its data, backup and recovery of the database, administration of the database and the need for a facility to handle non routine queries and reports. If successful implementation of a database for general use within an organization such as AFIT is to be accomplished, these items are aosolutely essential and must be carefully addressed before the first piece of data is assembled or the resulting database will not be as effective or useful as it could be.

1.2. SOLUTION AND APPROACH

The purpose of this thesis is basically to pick up where the 1980 thesis project left off, plus expand it to cover other essential areas not previously addressed. The design of a consolidated integrated database suitable for use by all departments within AFIT, a recommendation for a host DBMS, and the design of an information system suitable for fulfilling the requirements of AFIT/CI is the desired end product. Because the concept of a central database has already been recognized and accepted within AFIT, now is the time to tackle and solve the additional issues associated with such a project.

Some replication of the effort involved in the original thesis will be necessary because both of the schools originally interviewed must be contacted again in order to revalidate their data requirements. Also, the departments responsible for non-resident programs and those departments not originally contacted in the first survey will be given the opportunity to include their data requirements. Only after this is completed will there be a solid core of data elements from which an integrated database can be built and effectively implemented.

Concurrent with the above actions, steps will be initiated to review all available literature on database management systems and their capabilities. Each department will also be asked to state what they want from a database management system and what they would like to be able to do with one. These desires will be combined with the functions required to perform basic database manipulation tasks and used as the criteria for a critical analysis.

As many DBMSs as can be located and are commercially or currently available within AFIT or the federal government will be investigated. A predesignated set of criteria will be established with which each DBMS will be judged based on available information. Criterion will include such things as availability, cost, compatibility with existing hardware, reliability, backup and recovery features, security and query capabilities. Each DBMS will be assigned a set of weighted values according to how well it fulfills each item. The results of this evaluation will then be ordered according to the totals compiled for each database system with the highest scoring ones being considered primary candidates for implementation.

The results will be documented, included in this thesis writeup and also presented as a set of formal recommendations to the AFIT Computer Configuration Board (CCB) for their consideration and decision as to which system should be utilized. The emphasis is on finding the DBMS or DBMSs from all those currently available, either in-house or commercially, which can accomplish the job in the most economical and efficient manner.

Included with these recommendations will be a normalized database schema which can be directly implemented. This structure will insure minimal data redundancy, provide ease of access and preserve as many of the data dependencies as possible while maximizing data reliability and integrity.

Should a DBMS which is not currently available be chosen, the overall implementation process of AFIT/CI requirements will have to be amended. If this situation arises, an available system which closely matches the chosen one's architecture could be used to develop as much application software as possible. In any event, a fully documented "paper" design of the applications programs, including a Functional Description (FD), detailed program Specifications, and all data requirements needed for inclusion into the database will be completed.

The applications programs to be designed will provide AFIT/CI easy access to the database on a daily basis, thus beginning the move toward the replacement of the current and outdated manual system. A routine query capability, the production of standard reports and documents and the ability to answer unusual or ad-hoc questions initiated by higher management are among the desired, and currently unavailable, capabilities.

The design of the application system will be based upon the techniques of structured analysis and design. This will insure a complete top-down design and a total understanding of their operation so that the person(s) who follow this thesis with the implementation of the database and writing of the applications programs will be able to code, and easily maintain the system as required.

Of prime importance to this project is not only a complete AFIT database and a comprehensive design of the AFIT/CI applications programs, but a complete set of documentation for the procedures used to select the DBMS. It is hoped that this thesis will serve as a guide to others who wish to choose and implement a database management system where none currently exists, as well as be an exemplary use of software engineering techniques for the design of any applications programs.

1.3. ASSUMPTIONS

There are some important assumptions embodied in this thesis. They are:

(1) The application of automated database techniques offer the optimal solution to both the long and near term AFIT situations.

(2) The information management problems of AFIT are real and require a timely and adequate solution in order to meet mission requirements.

(3) Currently available, off-the-shelf hardware and software can be used. Some of which may or may not currently reside within AFIT.

Assumption one is the primary motivating force behind this thesis effort. The fact that an information management problem exists and has been recognized throughout all levels of AFIT is evidenced by some key events. These include the initial AFIT database thesis by Lt. Allred (REF 4), successive establishment and work on the AFIT/EN database by Dr.

Lamont, the current emphasis by the Computer Configuration Board (CCB) and the Computer Resources division (AFIT/ACD) concerning information management, plus the comments received from the faculty and staff.

A major assumption in this thesis is that automated database techniques offer the optimal solution to AFIT's near and long term information management problems. In this context, "automated database techniques" is used to mean generalized database management software designed to run on a computer, as opposed to non-automated techniques such as manual filing, or certain special automated systems such as file management systems.

While the distinction between automated and manual "database" methods is obvious, the distinction between database management systems and file management systems is not. In general, a database system is data oriented while the traditional file system is function oriented. File management systems directly relate fi'es to specific programs. Their arrangement, distribution on storage devices, and organization are developed solely to achieve optimal program performance. Data is normally accessed via an application program only and the only growth which occurs is in terms of data volume. Unfortunately, this is a dead end situation because data in one file and for one set of programs is generally not available or compatible with other programs. Also, common data in several files produces considerable data redundancy and data updated or changed on one file will not necessarily be updated accordingly in the other files which also contain it. (REF 7)

A database, on the other hand, is designed to provide generality, flexibility, and extensibility, both in the representation of the various records and in the files which comprise it. A database is meant to be a dynamic information resource to be drawn upon by a growing and changing community of users and not a preestablished set of files with rigid format and a fixed relationship to the application programs. (REF 7)

Although it seems clear that a Management Information System (MIS), which has a database as its basis, can be a logical solution to an information management problem, one must be sure that there are sufficient economic rewards to offset the associated costs in manpower, software and hardware. Unfortunately, there are no set equations which can be used to quantitatively measure the potential that a database holds for a given organization. However, some general situations have been identified by industry which point toward the desirability of a database in specific situations. In general, situations in which on-line inquiry capability is required, and where there will be growth in data volume and/or data types and a corresponding wide community of interest in database applications, indicate the potential and desirability of a database system. Currently all of these factors are present and readily identifiable within AFIT. In addition to these general reasons for using automated database techniques, there are some specific advantages to AFIT. (REF 8, p 459)

The first and major advantage of a DBMS is that of real-time, online data accessibility. Routine queries, reports and ad hoc queries can be performed interactively by non-programmers in minutes rather than the hours or days normally associated with manual or less automated methods.

(REF 15, p 425) In addition to responsiveness, another aspect of a DBMS is the existence of a complete query capability. This means that a DBMS has the ability to answer complex questions using both primary and secondary data characteristics, plus perform certain calculations and logical functions on and against the data according to the needs of the individual user. (REF 12, p 35)

A second advantage of DBMS's is their potential to minimize cost. This includes minimizing total data storage requirements, software and data redundancies, programming efforts due to data changes, training costs and the overall amount of paper associated with other automated systems. Storage requirements are minimized in two main ways. First, by consolidating all common data formerly replicated in various places, and second, by physically representing data in storage differently than the logical representation which the application programmer or end user requires. Software redundancies are minimized because the functions of data collection, verification, retrieval, security, storage and maintenance are all handled by the DBMS instead of on an application by application basis. (REF 15)

Reprogramming efforts due to data changes are minimized because of data independence, limited accessibility, currency and consistency. Data independence means that each application program logically views the data in terms of its own needs and may be entirely independent of, or isolated from, those used by other applications or users. Limited accessibility implies not only separate logical views of the data but a set of limitations on what functions can be performed on the data which may be common. This means only certain users have the right to create, modify or destroy certain data, thus preventing unauthorized applications or personnel from changing or damaging important data, either intentionally or inadvertently. Currency refers to the fact that alterations in the actual physical file are performed only by the DBMS. When it is determined that a change should be made in the data or in its logical or physical representation, the DBMS, under the direction of the Database Administrator (DBA), performs all appropriate changes. The results are more current and accurate data for the community of users which need it. Training costs are reduced because the existence of powerful user languages available on many DBMS's means people are no longer required to have expensive and lengthy classes in programming and computer science in order to use the database. These languages are designed to permit relatively untrained users to query and update data in a database, manipulate data, and generate reports in specific formats as dictated by their needs. Finally, paper usage is reduced tremendously because data traditionally "stored" on large printouts is now left on the computer and only called out when needed. A DBMS also allows users to only access the exact portion of the entire database they wish to see, thus freeing them from wading through mountains of reports and other useless printed data. (REF 12, p 32-34)

A third advantage of a DBMS is its ability to represent the inherent structure of the data by logically modeling the relationship(s) which exist among data of differing types. This translates into an ability for each user/application program to have its own view, which means everyone can see the data in the most natural and easily used form, and not have the format dictated by the limitations of the computer. (REF 15)

A fourth advantage is the ease of growth. Many DBMS's allow for easy restructuring of the database as new data types and new applications are identified. Restructuring of the data is possible often without rewriting existing applications programs or affecting other data which do not utilize the restructured portion of the database. (REF 15, p 23)

A fifth advantage of DBMS's is the inherent ability to use integrity checks. This means that although the database contains data employed by many different users, the data items and associations between data them will not be destroyed. Other integrity checks involve insuring that data values conform to certain specified rules, e.g. are constrained to lie within certain ranges or values. (REF 5)

A sixth advantage of a DBMS is tunability. Because real-time/on-line data access is such a key consideration, the time it takes a system to respond is of vital importance. Once the database has been established, users will begin to want different services from the DBMS as they become familiar with the many ways the system can be used or as applications programs evolve. Such changes can have a major impact on the organization and storage of data and ultimately on the response time of the DBMS. Therefore, the ability to tune or improve the performance of the database is very important. (REF 15)

A seventh advantage of a DBMS are the ancillary benefits that accrue to organizations which utilizes them. Since the creation and administration of a database is channeled through one agency, the DBA, the database itself is subjected to standardization. Just as in a standard language, standardization of a data simplifies its use, instruction, interpretation, problem formulation and programming. (REF 12, p 35)

Security is another benefit, which to some organizations is the most important. Besides the authority required by the user to access the database in general, there is also specific authority required to use a given logical view. Also a further level of protection inherently resides in the logical view itself, thus restricting portions of the database from unauthorized users. Additionally, in some DBMS's security controls can be even placed on individual data elements. It must be emphasized, however, that no system is totally secure, but the above measures do exist to make life as difficult as possible for persons who would unintentionally or by design attempt to compromise the security of the database. (REF 12, p 35-36)

The conditions which make the use of a DBMS advantageous plus the general desirability of such a system are present at AFIT right now. This fact, coupled with the power and many advantages of DBMS's serves to support assumption number two. Assumption three is based on the power, sophistication and advantages of current DBMS technology. Given the current state of the art of off-the-shelf DBMS systems, the cost/benefit ratio of using an existing system versus developing one in house is heavily in favor of using an existing system.

1.4. CURRENT KNOWLEDGE

(1) Previous work on a consolidated AFIT database has been performed, primarily within AFIT/EN and AFIT/ACD, and will be used as a starting point.

(2) Some preliminary data concerning required outputs, reports, data elements and procedures has been accumulated by AFIT/CI personnel.

(3) The AFIT/CI project has been entered into the AFIT ADP planning process and resources are available for commitment to this project.

(4) Commercially available systems (both hardware and software) presently exist which can satisfy this requirement if configured correctly.

(5) Data security and backup and recovery for the overall system is a primary concern and must be adequately addressed.

(6) Some attempt has been made on the part of AFIT to consolidate the information requirements of all of its directorates.

II DATA REQUIREMENTS ANALYSIS

2. PURPOSE

The purpose of this requirements analysis is to gather as much data as possible from each of the AFIT directorates and their data systems, gain a thorough understanding of what information they need to do their job, and consolidate it into the smallest functional set possible. Additionally, an attempt will be made to formulate some conclusions, based on the total bytes of data, as to the amount of memory which would be required to allow the database to function acceptably. The data to be analyzed will include individual data elements and their size, all appropriate reports/documents, information sources and destinations, and departmental interfaces for all of AFIT.

This analysis will be an expansion of the limited one conducted as part of the previous thesis. (REF 4) However, this time, all of AFIT and not just the Schools of Engineering (EN) and Logistics and Systems (LS) will be included. It is intended that the results of this study will provide the most complete and concise set of data requirements yet compiled. Once this has been accomplished, the results can be used to project the minimum processing support (hardware, system software and memory) plus the type and functions of possible database management systems. When considered together, these factors will provide a database with sufficient access, protection and utility to be truly useful at AFIT.

By gathering the types of data described above and combining it with the frequency of use, security requirements and how it is used, a clear and accurate understanding of the function, objectives, responsibilities and procedures of each department can be achieved. This will aid the AFIT Computer Configuration Board (CCB), the group which has been tasked by AFIT/CC to implement a database management system, in the completion of this goal. It has been requested by the chairman of the CCB that this analysis be supplied to the CCB for their consideration and action when completed.

If the individual departmental data requirements cannot be defined or the requirements remain ambiguous, the resulting database will not be able to solve any of AFITs problems, but merely duplicate them in an automated environment. Therefore, the accuracy of this analysis is vital, not only to this project, but to the AFIT goal of implementing a Consolidated AFIT Database and Information System (CADIS).

2.1. PROCEDURE

In general the procedure to be followed during this analysis will be one of seeking out and gaining an understanding of any information which presently exists within AFIT which might be applicable to a consolidated database. This material will be studied to find out how it correlates, and then arranged into a format which is easy to comprehend and handle. The final step will be the expansion/shrinking of this list by including the requirements from each of the departments throughout AFIT.

A thorough working knowledge of any data requirements which currently exist is important and must be achieved prior to the solicitation of any new data. This will offer some good clues as to the intentions and desires of the institute and eliminate numerous wasted hours in rediscovering old material. It is already known that at least two AFIT departments have some kind of studies, reports, etc. on hand which might pertain to general data requirements. AFIT/RR is completing the implementation of a commercial data management system specifically designed to perform registrar functions. They have defined the data elements they require, set up the appropriate files and are completing the applications programs which will support their operations. Additionally, the AFIT Data Automation Division (AFIT/ACD) has accumulated some general information concerning the general data requirements which should be included in an institute wide database. Available data from these sources will be collected, reviewed for completeness, consolidated and used as a basis for establishing the total data requirements for all of AFIT.

Once the on-hand data has been collected and restructured into a more readable format, the process of presenting it to the remainder of AFIT, in order to obtain any additions/deletions, will begin in earnest. The heads of each of the directorates and schools will be asked to supply any information they can which would lead to a complete definition of the data requirements specific to their departments.

In addition to the departments not considered in the original thesis project (REF 4), AFIT/LS and AFIT/EN will be asked to revalidate their previous input. Also to be included in this analysis will be AFIT/ACD and AFIT/CAE. During the meetings with the directorate heads, or their representative(s), the contents of the EN database and RR registrar system will be explained and discussed. It is anticipated that in so doing, common areas may be discovered as well as new ideas identified.

Of primary interest is the data elements/information, already in the EN database, which each department can use "as is", without modification; and that data which can be used if modified (if so, what needs to be done must be spelled out). By identifying these elements early in the analysis, the factor of commonality can be used to its greatest advantage. The earlier this can be done, the less time is wasted in research and meetings, plus the final product is enhanced and made more efficient.

Secondarily, each functional area will be asked to supply all appropriate information pertaining to any data elements/information they require but which are not currently accounted for. This input will include the sources and destinations of their data, the major reports/documents used, the frequency of data access and a complete description of all the data elements which compose them (this includes definition, size and functional dependencies). Another item which is extremely important and can only be supplied by the departments is an estimation of the quantity & size of each piece of information (or record). This will to be used later estimate the overall size of the database.

data gathering has been accomplished, the results will be reduced into the smallest "functional" or useful set possible, any common (or near common) reports/documents identified and the major information interfaces shown. Each data element will be listed alphabetically, and by record, along with a description and its length for later use. In order to enhance the utility of this information, a set of tables (relations) will be constructed, which represents only one possible database schema, and presented to the CCB.

2.2. RESULTS and CONCLUSIONS

The process of conducting an in-depth requirements analysis has provided a great deal of data and knowledge about both the overall and specific information requirements of AFIT. In order to adequately present this enormous amount of information, gathered from interviewing approximately 40 people over a four month period, the data has been grouped into some general categories. These categories include the basic assumptions and ground rules which governed the interviews, the actual results of the interviews, any unidentified or incomplete data requirements, recommendations for the use of certain data and structures, recommendations for the management and organization of the Consolidated AFIT Database and Information System (CADIS), an implementation plan, possible interfaces to CADIS, and a discussion of the overall structure and format of the database itself.

2.2.1. General Assumptions of the Data Analysis

The basic assumption, as has already been stated, is that the previous thesis by Lt. Allred (REF 4) would serve as a basis or starting point. By using his results it would be possible to quickly gain a great deal of insight into the nature of the requirements for the remainder of AFIT. Another assumption, and one perhaps just as basic, was that a single consolidated and integrated database would best serve the needs of AFIT. This does however, include a distributed database, or one whose data resides on more than one computer and has a single structure and definition, but specifically excludes multiple, independent database systems.

The general guidelines adopted in regards to the overall conduct of the requirements survey and analysis were designed so as not to impede or restrict the type or nature of the input. Users were not questioned as to why a particular piece of information was needed, only how it would be used. Therefore, all requirements were considered valid for possible inclusion the database design. This ultimately resulted in an analysis which was of a greater depth than anticipated and more inclusive than any done previously.

The other procedural guideline which was used also concerned the data requirements. Requirements which were found to be incomplete, not clearly defined or beyond the scope of the final project could not be included in the database design. However, in the interest of completeness and accuracy, all such requirements are listed separately, along with who identified it and an explanation as to why it was not included. Therefore, those who follow-up this project with a full scale implementation will have some idea as to the areas which require further attention because all the AFIT information requirements now known will be recorded in a single place with no question as to who needs what.

2.2.2. Results of the Surveys

On 3 July 1982 the AFIT Computer Configuration Board (CCB) officially asked that the results of this project be formally presented to them. Dr. Wolaver, AFIT/NR, in turn signed a letter on 16 July 1982 (Appendix IA) which officially announced the impending analysis and requested the cooperation of all directorates. This letter in effect granted the authority necessary to allow an AFIT wide data requirements survey and analysis. As a result all the AFIT directorates and schools were contacted, interviewed and provided an opportunity to submit their individual requirements.

2.2.3. Synopsis of the Procedure

Using the data elements identified in the previous thesis (REF 4), along with those already being used within the AFIT/RR registrar system, a worksheet was prepared for use within the interviews (Appendix IB). It was felt that this would aid people by serving as a "mind jogger" and give them a general idea of what was available, what was possible and what was being sought. In the course of ar interview, each directorate representative was given a brief background of the project and a summary of relevant past events. The worksheets were then presented and their content and intent explained. Participants were requested to use the sheet as a "scratch pad" and signify those data elements and relations they could either use with no changes, use if modified as they so indicated or had no need for. Additionally, they were asked to somehow identify any new information which might be peculiar to them and would be needed to make the database a useful utility for their organization.

Each department was requested to be as expedient and complete as possible but encouraged to take as much time as was necessary to fully cover their directorate. The individual results are listed in Appendix IG by respondent.

2.2.4. General Comments on the Interviews

The reception by the vast majority of people contacted in regards to this project was enthusiastic and supporti·e. However, most expressed frustration and dissatisfaction over the inability to maintain and effectively use the information required by their organization. Current functions and responsibilities, plus current and projected requirements and problems were all discussed freely and openly in a professional and receptive manner.

Almost universally, it was expressed that CADIS was desperately needed and that once implemented, would go a long way toward alleviating some critical problems, assuming certain principles were kept in mind. Foremost of these was that such a system must provide easy access to the database plus the ability to conveniently query and manipulate the data as need be. Applications programs and their subsequent support was generally not desired because an adequate DBMS could give them greater flexibility through the query facility and report generator without the cost of software development and maintenance. When discussing the need for reports and other required documents, it was found, as expected, that if these facilities were present, the need for many paper records and inter departmental reports would vanish.

The School of Engineering (AFIT/EN) and the School of Systems and Logistics (AFIT/LS) were interviewed in greater depth than other areas because some of the basic data elements used were a result of the previous in-depth analysis. (REF 4) It was felt that the individual department heads should be contacted so they could revalidate their overall requirements. In most cases this resulted in a great deal of new information as well as modifications to others.

2.2.5. Overall Data Requirements

The purpose of this section is to present an overall picture of the results of the data requirements analysis. They will be discussed in general terms but are more clearly and accurately presented in chart form in appendix IIC.

The overall goal of this project was to identify all the data requirements and design an integrated database which would reflect all the current needs of AFIT. In reality however, some directorates presented data which they did not elaborate upon nor was there time to define completely. Therefore what was actually gathered was a solid core of highly common data, which if implemented in a timely manner, should satisfy a very large amount of AFIT's information requirements. This reasoning is based on the fact that almost all of the requirements expressed revolved around a sub set of the overall data. This "central core" consists of general student, faculty and staff data plus several smaller relations which provide necessary and more detailed additional information.

Another goal of those interviews was to gather a list of all the reports and documents which were used within AFIT. However, this was found to have been completed by AFIT/ACD along with data flow diagrams for the entire organization. These may be found in the 1981 AFIT ADP Master Plan (REF 3).

The overall structure of the database can be found in the form of a diagram in appendix IC. This chart represents each of the relations identified to date plus a depiction of the most important logical associations between them. A line connecting two relations means that there is at least one common attribute between them and that there is also a meaningful logical connection between the two. All student and staff data is accessed by the Social Security Account Number (SSAN) of the person of interest.

Many attributes in these relations are codes or short names, but a user can obtain full titles or expanded information on most of these codes or abbreviations. This is accomplished by using a logical connecting code and retrieving the associated full description, or in some cases by referring to the data dictionary.

The chart in Appendix IC denotes no other meaning than that described above. The size of a figure, nor the length, nor the number of lines implies any information.

More information on the content of the database can be found in appendices ID, IE, IF, IG and IH. The first four documents essentially comprise a highly detailed data dictionary, while the fifth shows the database size estimates. Appendix ID gives a functional description of each of the relations on the chart along with the length of each tuple. Appendix IE is another list of relations but this one shows the attributes which comprise each relation with the key attributes for each, denoted with an "*". Appendix IF provides a detailed breakdown of all the attributes by listing the size of each attribute a description, an example of how it is to be used, a list of the relations which incorporate it, and any synonyms it might have. Appendix IG is a list of the logical views of the database. These views also reflect each directorates data requirements as presented in the interviews. Appendix IH gives, for each relation, the estimated total number of records stored in memory at the end of the first years operation of CADIS. Also given are the factors going into each estimate and the major source of those factors.

2.2.6. Database Size Estimation

The estimated size of the database is 28,550,000 bytes of data. This figure represents the total amount of memory required for one full year of operation (several relations maintain four quarters worth of data). No student history, other than a short educational history for 1000 students, or registrar data has been included in the estimate.

This estimate was derived by taking the size of each relation, as determined by the sum of the lengths of its attributes, and multiplying it by the estimated number of tuples for that relation. Figures for the lengths of each relation were derived by actual counts, estimates from faculty or staff members or by the authors' best estimation and can be found in appendix IH.

For ease of computation it was assumed that all data would be stored in character format and that one character would occupy one byte of memory. This size estimate does not include the core memory required for the DBMS itself to function.

2.2.7. Undefined/Incomplete Data Requirements

Appendix I encompasses the entirety of the AFIT information/data requirements identified during the interviews. Implementation of the database as designed will alleviate approximately 90-95% of the AFIT information problems identified. The data considered as "missing" from the design was compiled from two entirely different sources, the worksheet and comments by each interviewee, and represents those items which could not be included into the design for the reasons listed below.

The reasons this data was not included is because it was: a) submitted too late in the design process, b) stated in a format which was much too general in nature, or c) contained various elements were very specific to a single area and not desired by anyone else.

The largest single influencing factor concerning the information not included was that of time. Each interviewee was contacted in sufficient time to respond with their requirements and have them included (some were given much more than were others). However, there were those who were unable to identify their total requirements quickly enough. Therefore, only that portion received early enough was able to be included, In general, the data which falls into this category is not to be considered "essential" and the current structure will fulfill the vast majority of the requirements.

Each person interviewed was asked to be as specific as possible as to what data was required for their area. Everyone was asked to use the worksheet to mark their input and any item which was not SPECIFICALLY identified as UNDESIRABLE was included. In spite of this, some people simply identified their "general ADP/information requirements" in a very non specific manner. For instance, the most common area identified this way was "budget data". Because of the nature of this thesis, there was insufficient time to perform the in-depth system analysis required to translate this type of request into actual database requirements. AFIT will have to supply the qualified personnel to work with these areas and perform this lengthy analysis at a later date.

Finally, some areas supplied data which was rather narrow in use and was oriented to a small subset of their directorate (mainly AFIT/EN and LS directorates). This type of data was excluded because it was felt that because it was not actually "common", it should be reviewed for desirability and applicability across the directorate prior to inclusion.

The data listed in appendix IJ is reproduced exactly as it was received from the worksheets.

2.2.8. Recommendations for Data Use and Data Structures

Because of the uncertainty as to which DBMS will eventually support CADIS, the structure and use of several attributes within the database may have to be modified from their format as presented in appendix I.

The primary factor leading to this decision will be the sophistication or limitations of the DBMS and its query facility. As an example, the attribute 'Course' may need to be broken into four smaller attributes Course_Type, Course_Level, Course_Number, and Section (e.g. EE686A = EE-6-86-A). If the components cannot be accessed directly, such elementary questions as "how many graduate courses (Course_Level > 500) are offered" or "what courses have multiple sections being taught (Section is non-blank)" cannot be answered because the DBMS might view the attribute as one contiguous, unbreakable character string. If, however, the query facility has a string search capability (e.g. select data when the third character > 5) this kind of breakdown would not be necessary. The ability to perform these operations will vary with the DBMS and must be closely studied prior to implementation.

The attributes which may require further breakdown are:

(1) Course; into Course_Department, Course_Level, Course_No, and Section.

(2) Office; into Directorate and Division (ENA = EN-A).

(3) Extra_Duties; into some more meaningful breakdown as determined by AFIT/DA.

(4) Section; into Section_Type, Section_Year, and Section_Month (GCS82D = GCS-82-D).

(5) Name; into Last_Name, First_name_MI (This holds for all attributes which contain a name like attribute).

(6) Date; into Day, Month and Year.

(7) Degree_Code; into Degree_Level and Degree (e.g. MS-EE).

Two additional pieces of data will have to be examined further and decided upon prior to the establishment of the database. These are the establishment of identification numbers for foreign students, similar to SSAN, and how much of the GRE and GMAT scores are to be stored. Initially, the database has been designed to carry total scores only, but if the demand is great enough for the component scores, it is recommended that a set of indicators and relations be established similar to those in the registrars relations (appendix ID & IE, relations 78 - 80).

It was a general rule to make all codes as meaningful as possible when viewed by themselves. Specifically avoided was the unclear or ambiguous use of single letters or digits. By so doing, a large amount of confusion will be eliminated when data is viewed or used apart from the full descriptions or long versions of the attributes. For example, a tuple such as "John Jones, M, 5135B, P, SE" tells its user that John Jones is married, is a 5135B computer programmer, is a pilot, is married and has a secret security clearance much clearer than does a tuple such as "John Jones, 1, 5135B, A, 1, 2".

Whenever possible a code was indexed into another relation which contains its full description or title. This practice makes adding or deleting codes from the database easier and also facilitates edit checking and validation. There are some codes used in appendix IF which do not use such a structure. In these cases the codes have a very limited range of possible values, say two to four. If a DBMS is chosen with the proper capabilities, these values can be defined in the database definition mechanism as parameters, and the DBMS softwar will perform edit checking automatically. When available, this feature is easy to use and change and frees application programs from time consuming and lengthy validity checks which must be performed in order to guarantee the integrity of the data.

All attributes are defined as characters for ease of computation and there are some distinct advantages to leaving the database this way. Depending on how much numerical manipulation is performed, the savings in memory and processor time might outweigh the computational speed gained by storing numeric values versus storing attributes such as No_Students as characters and letting the DBMS convert the values whenever computation is desired. This is a decision which must be left to those who will implement this design.

2.2.9. Recommended CADIS Management Structure

Volumes and volumes of material have been published over the years on the technical aspects of DBMSs. On the other hand comparatively little has been written on the subject of management techniques, practices or methods associated with implementing and running such systems. According to what literature is available it seems to fall upon the individual organization to review their situation and develop their own methods. There is however, a set of sound, concrete reasons for going to the trouble of developing an adequate management structure. "The database will have a major impact on the enterprise, not only in terms of benefits but in terms of problems. Its fundamental aspect of the sharing of data among users and of greater central control of data are just the type of changes that can have severe political repercussions within an organization." (REF 16, p 5-1)

In order for an integrated database to be effective, such things as setting standard definitions, formats and usages for all data elements plus the associated monitoring and enforcement of these standards is essential. This will insure the existence of accurate and consistent data for all users who will quickly come to depend heavily upon the database for meeting their mission requirements.

"Resentment to the disciplines imposed by the database can often arise; some departments might insist on applying their own standards [or methods of database management]. Such difficulties can be overcome, but only if the continued and dedicated support [and guidance] of senior management is assured." (REF 16, p 5-1)

The solution most often adopted, and recommended by industry practice and writing, is to establish a specialized area of the organization whose primary consideration is the administration and control of the database and its operation. This 'Database Administrator (DBA)' must be "familiar with the use and nature of all the data, ...be vitally concerned with the performance of the system and...be able to negotiate with and resolve conflicts between user departments...". (REF 16, p 5-24) In short this position should be filled with someone who is totally competent in all the technical aspects of a DBMS plus have intimate knowledge of the organization and its operation. But most importantly, he must be someone who has the authority to make and enforce decisions involving the database.

2.2.10. AFIT Management Features & Problems

In most respects AFIT is no different than other civilian and government organizations. There is a shortage of qualified personnel and those who are available do not have the necessary database experience to implement and control such a project. For the most part AFIT will cope with these universal problems just like everyone else and rely on the available pool of expertise from other areas and accomplish the project anyhow. Additionally, there are some problems which while they are not unique to AFIT, present some real obstacles.

There is a decided lack of practical experience with the design, implementation and management of DBMSs. This, coupled with the fact that this would be a "new start" project could combine iuto a deadly combination if the proper precautions are not taken immediately.

AFIT has a diverse community of users which view the concepts of database systems and data automation from three extremely differing perspectives; that of academic, administration and real world. The result is a situation not too uncommon, that is, differing opinions on the control of data and the methods and procedures for implementing a centralized database.

2.2.11. CADIS Management Recommendations

The optimum solution for the administration of CADIS appears to be one with two components. First, "the solution most frequently offered for this type of situation is placing the administrative responsibilities in a staff position, reporting directly to the highest manager responsible for data processing in the organization." (REF 2, p 6) Secondly, "the database administrator functions should be a team rather than a single manager". (REF 16, p 5-24)

The Database Administrator (DBA) position is a very demanding one. He is faced with deciding such things as "resolving conflicts of ownership if several departments wish to amend or retrieve the same data...negotiating with users to establish who has the right to access or change data items...providing recovery facilities and establishing precautions for applications in the event of a breakdown...establishing data entry, edit and validation standards, and maintaining the data dictionary." (REF 16, p 5-26,27) "For the data administrator to have sufficient authority to be accepted by user departments, to reconcile their conflicting requirements for data, and to be able to enforce standards, the position must be relatively senior within the enterprise." (REF 16, p 5-26)

Based on the above comments it is recommended that AFIT initiate two separate levels or areas of administration for CADIS. The first, that of the Database Administrator (DBA), should be a senior member of the staff with sufficient authority and knowledge of the organization to shepherd the database, with all of its growing pains, through to full adulthood.

To aid the DBA, and as part of the first level of CADIS administration, each directorate should have a single representative or Data Administrator (DA) which would represent the views and needs of their respective functional areas. All new requirements and problems would first be reviewed by a DA for clarity, consistency and practicality. He should also be able to answer any question about current data usage and meaning as well as act as the focal point for any problems or changes to the database originating from or directed to his directorate. Once discussed at this level, anything of interest to the community of users, valid changes to the structure of the database or disputes should be taken to the DBA for review and action. The DBA would check them for validity, then for consistency against prescribed rules and standards and act accordingly.

Once the DBA has approved the changes or resolved any problems with the recommendations, they would then be passed on to the other component of CADIS management, the Database Manager (DBM). This person or group would act only on the direction of the DBA and should not be a member of the staff but someone with a more technical orientation toward DBMSs. This must be the case because he is the one who will monitor and be in charge of the DBMS software, make any changes to the structure of the database as dictated by the DBA, as well as being the interface to the vendor and acting as the ultimate in-house technical representative. The DBM should not be involved in data oriented problems, but only in the efficient operation of the DBMS itself on whatever computer system it is installed.

It is very likely that the CCB could suffice for the first level of the previously mentioned structure, with the directorate representatives, or their appointees, acting as the DAs. The second level of administration should be performed by AFIT/ACD. They could supply one or more people to act as the DBM since the type of the technical expertise desired resides with them. The job of DBM would very likely be a full time assignment during the infancy of CADIS but could become a background duty once it is operational.

In general, each functional area needs to be responsible for their own data, AFIT/ACD should be in charge of the computer and the DBMS software and serve as technical representative for their combined use.

For this to become a viable management framework, the general level of education in regard to DBMSs has to first be increased. AFIT should immediately begin to send people to the database courses (basic and advanced) offered within the School of Engineering. They also need to talk to vendors and visit other AF installations which have experience in database systems. Additionally, the heads of each of the directorates should at least attend the database short course available periodically at AFIT/EN. These suggestions are wholeheartedly recommended because the uniqueness and nuances of DBMSs dictate a need to become familiar with the theory and practices of a database operation. Without this basic understanding of the functions and capabilities of a DBMS, by all those involved, it is nearly impossible to successfully implement CADIS and have it as useful as it could be. Such education should be above and beyond the normal training classes which will be offered for the DBMS by the vendor once it is installed. The DBA and the DBM should be the first to seek this type of knowledge, and as soon as possible, followed closely by the DAs and their users. One can never hope to build and operate that which he does not understand and trust.

2.2.12. Database Implementation Recommendations

In order to implement a database and bring the users 'on-line' with the minimum amount of mayhem and confusion, plans must be drawn up for the implementation well in advance of the actual event. The purpose of this section is to suggest one scheme which would ease the 'labor pains'. A basic premise for any implementation plan is to try and achieve the most results with the least expenditure of and resources.

First, a Database Implementation Team (DBIT), composed of at least the DBA and DBM, should evaluate the user community in order to determine any highly critical areas which were not known at the time of the requirements analysis and superimpose them upon the following general plan discussed below. Additionally, AFIT/ACD will have to supply two or three people on a full time basis for at least a year in order to implement CADIS.

2.2.13. General CADIS Implementation

The DBIT should not under any circumstances attempt to load all the relations prior to allowing the users access to the database. This is just common sense, good top-down implementation practice and will prevent massive confusion and eliminate wasted human and computer time spent resolving problems. If all the relations are loaded at once, this would cause the majority of the users, who could operate with a subset of the data, to wait an unnecessarily long time before becoming operational. Also, by giving the data to the users in usable segments, they will have the opportunity to fully evaluate and test the features of the system and find any errors in the data or its structure easier than if all of their data was present. Additionally, when a new system is presented, "complete" or "fully capable", the users tend not to learn it thoroughly nor learn how to use its full capabilities, but learn just enough to get by.

The inclusion of the registrar should be left aside at this time because of their current automation effort and the many complexities and controversies associated with their type of data. But if, at some later date when the all of CADIS has been implemented and exercised, it is deemed desirable, then and only then should AFIT/RRs inclusion be considered. What should be implemented at this time is the subset of the registrars data which was requested by the remainder of AFIT. By so doing it is assured that CADIS will be truly useful to its users. Also, the mistakes and hardships which surround an organization trying to operate with its essential data needlessly spread among multiple data systems are eliminated.

Finally, it would be easier all around, plus create experience with the new system if each user was generally responsible for loading their own data. No one knows the data like the individual departmentmental users do, and with the DBA and DBM assisting by supplying routines and advice, implementation would proceed much faster.

2.2.14. CADIS Implementation Plan

The intent of this plan is to install or expand (load and verify) the database in sections which will do the most good for the most people. The steps are not necessarily mutually exclusive and several steps may be in various stages of execution at any given time. More capable users may be able to get their data loaded quicker than others and no attempt should be made to slow them down if they can be administered adequately.

The following steps constitute the CADIS Implementation Plan:

(1) The DBA and DAs should mutually establish the data standards and codes which will be used throughout the system.

(2) Define the entire database structure in the computer at one time. This would consist of those relations and attributes, along with their rules, as defined in appendix I or by the DBA. By doing this first, modification of the basic structure can be kept at a minimum and nothing extra needs to be done when a user is ready to load a relation.

(3) Initially load those relations which comprise the 'core' of CADIS under the supervision of the DBA. This will allow AFIT/EN, LS, DE, PA, ED, and CI to go into limited operation very quickly and allow them to start testing out the structure and their data. This 'core' is comprised of the STUDENT and STAFF relations plus those relations which allow them to function properly. These core relations include: OFFICES, ACADEMIC_TITLES; STUDENTS, STAFF ED_CODES, SECTION, FOREIGN_STUDENT_DATA, CURRENT_THESES, CI_DATA; DE_DATA; RESIDENT_DATA; MAJCOMS, SPOUSES, LOCATIONS (only the subset large enough to allow minimum operations), and AFIT_CALENDAR.

(4) Once the 'core' has been installed and verified for completeness, the users can begin to load the special purpose relations which pertain to their particular operation. These relations include: BOXES, LOCKERS, ED_HIST, PAST_COURSES, ED_PLANS, EXTRA_DUTIES, EXTRA_DUTY_ROSTER, DUTY_OFFICER, PROMOTION_STATS, QUOTAS, PCE_DATA, DEGREE_SOUGHT, DEGREES and all the faculty related relations.

(5) Complete the loading of the LOCATIONS relation with the remaining data required by AFIT/CI and others.

(6) Finish installing the relations which pertain to AFIT/CI. These relations are: CURRENT_CI_PROGRAMS, CI_PROGRAMS, INST_PAYMENTS, INST_POCS, SCHOOL/EWI_DATA, MMEP_DATA, MED_BOARD_CERTIFICATION, MED_TOURS and MED_PROG.

(7) Load the course, book and room data. These relations consist of: AFIT_COURSES, PREREQ, COURSE_BOOKS, BOOKS and ROOMS.

(8) Load the course scheduling and other data from the registrars system. This includes the following relations: COURSE_TIMES, STUD_SCHED, SHORT_COURSES, FAC_COURSES, SHORT_STUDENTS.

(9) Load the leftover relations as the data becomes available.

(10) Begin loading the THESES and STUDENT_HIST relations as appropriate.

If this type of approach is followed, an effective management structure is instituted and the people are properly educated, CADIS can be at least 85% operational in about one year. Impacting on this estimate are the decisions as to the DBMS, the hardware and the speed with which the data is gathered and prepared for entry into the database. Preparation could begin immediately after the DBMS is decided upon by storing it on the machine to be used via tapes or a text editor. This data can then be easily prepared for final loading at a later date.

2.2.15. Interfaces to CADIS

For purposes of this section, the term "interface" will mean any data which one data system may provide or receive from another data system. There is no intention to imply a method or media for accomplishing the actual transfer.

The following are recommended interfaces to CADIS:

(1) Let the AF Standard Personnel System periodically supply data on the AF students and staff at AFIT. One problem identified by AFIT/DP was that the data maintained by the schools on their students is almost always more current than that maintained by themselves and the AF personnel system. By having AFIT/DP supply some attributes and have them updated only via the interface, the only way a particular students record could be changed is by going through AFIT/DP and modifying their permanent record. Changes in the data would then show up in the database at the next update. The primary reason for utilizing this interface would be to assure AFIT of the most current, official data on on a student. AFIT could therefore prevent their student data from becoming substantially different from his permanent record and visa versa. (NOTE: AFIT/DE currently receives a periodic data tape from Randolph AFB.)

(2) Allow the registrar to supply data to CADIS for the following relations: COURSE_TIMES, AFIT_COURSES, STUD_SCHED, FAC_COURSES, PCE_DATA, SHORT_COURSES, SHORT_STUDENTS along with the degree which a student is seeking. While this data probably does not exist in a directly transferable format, enough can be supplied and converted to guarantee consistency across AFIT.

(3) Let CADIS supply the registrar with the student grades at the end of each term. Instead of faculty members filling out grade sheets and passing them to AFIT/RR. Grades for students would simply be entered into CADIS once and the whole school could transferred in one easy step. Also, departments and advisors would have the past grades which they requested for each of their students.

(4) When the students schedule is received from the registrar (via interface number two) and entered into CADIS, a check could be made against his educational plan (ED_PLANS relation) and any deviations from that plan could be flagged for the faculty advisor.

(5) When the COURSE_TIMES data is supplied to CADIS, that same data could be used to update the ROOM_SCHED relation for the current quarter.

2.3. DATABASE STRUCTURE and DESIGN

"The real benefits of a database can be realized only if it models the operation of the enterprise. The database must represent entities and their relationships...The model must not be dependent on such issues as hardware, operating systems or host languages..[otherwise] the resulting design will not model reality." (REF 16, p 5-13)

"Relations can be used to model the "real world" in several ways; for example, each tuple of a relation could represent an entity and its attributes or a relationship between entities. However, in many cases, the known facts about the real world imply that not every finite set of tuples could be the current value of some relation, even if the tuples were of the right arity and had components chosen from the right domains." (REF 17, p 167-168)

These real world implications, which limit the tuples that can be values for a given relationship, point out the existence of certain restrictions on relations. Jeffrey Ullman distinguishes between two kinds of restrictions on relations. The first are those types of restrictions that depend on the semantics of domain elements. This class of restrictions tells little or nothing about the design of a database schema. Examples of this first type of restriction are that there is no (official) ACADEMIC_TITLE of "Tyrant", nor can anyone's LOCAL_ADDRESS be "Rural Route 1, Mars". (REF 17, p 168)

The second kind of relation restrictions are those that depend only on the equality or inequality of values of attributes shared by more than one tuple. This type of constraint does not depend on what value a tuple has in any given component, but only on whether two tuples agree in certain components. The most important of these constraints are called functional dependencies, and are a form of value-oblivious constraint. It is the value-oblivious constraints that have the greatest impact on the design of database schemes. (REF 17, p 168)

"Let R(A1,A2,...,An) be a relation scheme, and let X and Y be subsets of (A1,A2,...,An). We say X->Y, read "X functionally determines Y" or "Y functionally depends on X" if whatever relation r is the current value for R, it is not possible that r has two tuples that agree in the components for attributes in set Y". (REF 17, p 168)

One natural and frequently occurring example of a functional dependency is that of a key and an entity set. For example, if R represents an entity set, and A1,...,An are the attributes of that entity set, and if X is a set of attributes that forms a key for the entity set, then one can assert X->Y for any subset Y of attributes. This follows because the tuples of r represent entities, and entities are identified by the value of attributes in the key. Therefore, two tuples that agreed on the attributes in X would have to represent the same entity and therefore be the same tuple. (REF 17, p 168)

It is vitally important to realize that the declaration of a functional dependency in a database is a decision that can only be made by the designer. The advantage of such a declaration is that the database system will then enforce an integrity constraint for the user, and there could even be a more efficient implementation of the relation possible because the functional dependency is asserted to hold. However, there is a price to be paid, in that the storage of certain information becomes impossible. For example, if one decides NAME functionally determines HPHONE, then under no circumstance is it possible to store two home phone numbers for one person in the database. (REF 17, P 168-169)

(NOTE: In the foregoing and following discussion an important assumption is that a key X for R is by definition the minimal set of attributes that uniquely determines a tuple of R.)

2.3.1. Normal Forms For Relation Schemes

A number of different properties, or "normal forms", for relation schemes with dependencies have been defined. One of the most significant of these is called "third normal form" (3NF). This normal form guarantees that most problems of data redundancy and most insertion and deletion anomalies do not occur. (REF 17, p 187)

To define these terms, it is first necessary to define a "prime" attribute. An attribute A in relation scheme R is a prime attribute if A is a member of any possible key for R. If A is not a member of any key, then it is nonprime . (REF 17, P 187)

2.3.2. Third Normal Form

A formal definition of third normal form is: "A relation scheme R is in third normal form if and only if there does not exist a key X for R, a set of attributes $Y \subseteq R$, and a nonprime attribute A of R not in X or Y, such that

1. X->Y holds in R,
2. Y->A holds in R, but
3. Y->X does not hold in R.

If Y is a subset of X, and therefore by (3), Y is a proper subset of X, then R is said to have a partial dependency. If Y is not a subset of

X, then R has a transitive dependency. If R satisfies the above condition whenever $Y \subseteq X$, but not necessarily otherwise, then R is said to be in second normal form." (REF 17, p 187)

2.3.3. Summary of Design Approach

It was decided to adopt a relational approach to the AFIT database design for several reasons.

1. The power of relations to model the real world, both entities and their relationships.
2. The fact that the relational model forces the analyst to note the dependencies of each relation, and in arriving at the optimal third normal form, to eliminate many redundant items and to also eliminate many potential update anomalies.
   (REF 13, P 5-13)
3. The flexibility and easy expandability of the relational approach.
4. The simplicity of the view of the database that users see is that of two dimensional or flat files.

All relations in the AFIT database design (Appendix 1) are in at least 3NF. This was proven by an exhaustive inspection, i.e., by taking all functional dependencies for each relation (Appendix I,I) and evaluating them against the definition of 3NF. Any relations which failed to satisfy this formal definition were redesigned. This process was repeated iteratively until all relations were normalized to 3NF.

III EVALUATION OF VARIOUS DBMSs

3. GENERAL OUTLOOK, DESIRES and ASSUMPTIONS

Having already realized the need for improvements to the information handling procedures within AFIT and the fact that a DBMS is the proper and correct solution, it is now essential that the optimal DBMS(s) which can effectively accomplish this task be identified. The general purpose of this section is to examine as many commercially available DBMSs (including those already procured by AFIT and those available through the federal government), evaluate their features and characteristics and determine which one(s) will best support an integrated AFIT database as identified in the previous chapters.

The basic underlying assumption is the fact that the appropriate DBMS is currently available. There is no desire nor are there sufficient resources to invest in the construction of a system specifically tailored to AFIT. In reality the needs and desires of the personnel contacted are not beyond current technology and not substantially different from other institutions and organizations.

Extreme caution and care has been used in order to eliminate any prejudices, personal feelings or biases from the development of this evaluation process and its subsequent execution. The goal is to identify solutions without regard to personal preference; however, past experience and general knowledge will be an invaluable source of information and ideas.

The general desire of all parties involved in this project and of those who supplied input is that there be a general type of "environment" for the users of the system. It should be easy to use by both "programmers" and "non-programmers" while fulfilling the needs of each. The desire of easy access for AFIT personnel seems to show a trend away from the traditional application program interface for users. Based upon these feelings, one area of emphasis will be on "user friendliness" and the non-procedurality of the package.

A DBMS must have the inherent capabilities to provide sufficient flexibility to support a dynamic and growing organization like AFIT. The need exists for a true software "utility", that is, one which is reliable, dependable and not only meets current needs but can grow with the organization. "The capability of a database system to grow, in terms of the users and of the data to which they refer, is fundamental to the database system concept. This growth may be achieved in many directions and dimensions. The necessary requirement is that a data base package be able to accommodate modification, variation and extension of the database without requiring modification of all currently existing programs that use it." (REF 7 p 3.1.7)

An area of concern which is second to none is the ability of the DBMS to provide a level of security which can guarantee the protection of sensitive information while not unduly restricting access to those authorized to use it. Several departments expressed a need to safeguard their data in order to protect both its integrity and use. While only a single feature, the methods and degree which vendors choose to use in their individual implementation of security make this area one that should be investigated closely.

3.1. EVALUATION PROCEDURE and SHEET

As stated earlier, the evaluation sheet is used to tally or record how each of the DBMSs measured against a set of fixed criteria; which were determined by the needs of AFIT. The goal of this procedure is to evaluate all available DBMSs as objectively as possible by assigning a set of numbers or points to each system of interest. Once completed, the sum of these scalars will be used to "rank" the individual systems in order to provide a basis for the ultimate selection of a DBMS for implementation at AFIT.

By using the resources of the AFIT library, the General Services Administration (GSA), The Air Force Directorate of Computer Resources (AF/ACD) and others, a complete list of commercially available DBMSs, plus those available from within the federal government will be compiled. Based upon a set of "minimum" criteria the DBMSs will be screened to insure that only those systems which are minimally acceptable will be evaluated further.

Each DBMS will be measured against the items on the evaluation sheet (Appendix IIA) and assigned a point value based on how it meets the stated criteria. The vendors' literature will generally be the basis for these determinations. If there is insufficient data available, the vendor will be contacted and asked to supply what is required. Point values will be assigned by first comparing the DBMSs against four general rules. These rules are: 1) The DBMS exceeds the stated criteria, maximum point value is assigned; 2) The DBMS meets the stated criteria, median value assigned; 3) The DBMS does not meet the stated criteria, a zero (low value) is assigned; 4) The DBMS does not incorporate the feature at all, an appropriate negative value assigned.

Once this rough breakdown has been completed a finer evaluation based upon utility and effectiveness of the DBMS in each category will be performed. In so doing the first values (if any) can be adjusted up or down according to the degree to which the basic requirements are fulfilled.

After this process has been completed, an overall value for each DBMS will be computed and the systems ranked according to this total. The top DBMSs will be given as recommendations to AFIT for further consideration concerning acquisition and full implementation. It is from these systems that it is anticipated and strongly recommended that AFIT choose a DBMS package. Anything else might quite easily result in a system which cannot cope with the situation and requirements of AFIT. These results will be arranged in a tabular form and presented to the CCB as a part of the formal recommendations.

3.2. Description of the Evaluation Sheet

The actual evaluation sheet is composed of three major sections, each broken into detailed subdivisions. Each section corresponds to the areas of interest to AFIT and their relative importance is emphasized by a unique maximum value which can be assigned to each subdivision within that section. This type of arrangement was settled upon in order to insure that the "strongest" DBMS, i.e. best suited to AFIT's needs, would be identified. It is strongly desired that no single factor or group of factors should handicap (either positively or negatively) a particular DBMS. It is recognized that ultimately the final determination as to which DBMS is to be implemented rests with the Computer Resources Directorate (AFIT/ACD) and the commander of AFIT, and will involve factors not within the scope or nature of this project.

The first major section deals with those "real world" considerations of great importance to AFIT. Accordingly a maximum weighted value of 10 points per subdivision is used for emphasis; this shows that factors so fulfilled are a definite advantage to a DBMS. Areas of consideration include such things as machine compatibility, current availability and cost. These were chosen because they are of the type which will play a significant role in ultimate selection of the DBMS. Therefore, it was decided that they carry the highest maximum point value.

The second major section is concerned with the operational features of a DBMS. This section is designed to measure features which are important, but which it is assumed that every DBMS will incorporate in one way or another. Therefore, a maximum weighted value of eight points per subdivision is used to help distinguish to what degree they are available, how well they work and how easy they are to use (according to all available information).

The third major section deals with items which are available with some DBMS packages or would be "nice to have" but are not required specifically. It was felt that if a DBMS possessed any of these features it would be a plus but the lack thereof should not be a detriment from an otherwise desirable package. A maximum weighted value of six points per subdivision is used.

It is recognized that some of the items to be examined are more subjective than are others. Every attempt has been made to reduce these occurrences, but those which remain will be evaluated based on previous experience and the consensus of others with more experience with DBMSs.

3.2.1. High Priority Considerations (Section I)

3.2.1.1. Current Availability

The location or source of the DBMS is scored based on the ease of acquisition. It includes, from a high value of 10 points down to the low value of two points, DBMSs currently located at another WPAFB organization, another Air Force installation/facility, some other organization within the Federal Government and those commercially available only. DBMSs currently within the pervue of AFIT will score the maximum value.

3.2.1.2. Structure

The underlying data structures and associated operators the DBMS will support is a crucial question. Database systems are categorized according to the particular approach which is adopted in answering it. The three best known approaches, relational, hierarchical, and network will be used as a basis for analysis. (REF 10, p 63) In accordance with a significant majority of the academic and industry opinions and trends, a DBMS which is based upon the relational concept will rate the highest value possible (10 points) followed by network (5 points) then hierarchical (2 points).

3.2.1.3. AFIT Computer Systems Compatibility

Regardless of how good or how easily available a DBMS may be, if it will not run on a computer system currently at AFIT or projected for acquisition, then its utility will become largely academic. The greater the number of systems, at AFIT, which could host a DBMS, the higher the score for that system. Current AFIT computer systems include: CYBER 70, HARRIS, VAX 11/780, and UNIX on the DEC 11/60, 11/34 and 11/780. In order to aid in the office automation project at AFIT, any DBMS which operates under the UNIX operating system will receive bonus points.

4.2.1.3.

A database system requires core memory and peripheral storage space; not only for the actual data of interest to the users, but for the database software itself. For the system to operate successfully this implies a minimum hardware configuration and capacity. (REF 7 p 3-6)

There are three general questions which will be considered. First, can the DBMS concurrently support multiple databases? (4 points) Second, does it use the host computers file management system (access method); versus employing a unique one? (3 points) Third, is there a limit to the number of records or bytes which the system can handle before performance begins to degrade? (3 points) DBMSs which have no limit will receive maximum points with the actual limits determining any lesser value. Scores for DBMSs which cannot do this will be assigned values according to the particular circumstances. If the system does not operate with the most efficient access method available on the system, the score will be reduced accordingly.

One important consideration which cannot objectively be assigned a point value, but will be listed for each DBMS is the amount of storage/memory overhead required for the system to operate.

5.2.1.3.

No score will be assigned for this factor. The dollar figure will be listed along with the various options for acquisition, i.e. purchase, lease, etc. Any additional charges imposed by the vendor will be noted and included in later sections.

5.2.1.4. Machine Independence

The ability to enhance or change a computer system is vital to a growing and changing organization. Likewise the ability for the DBMS to grow/change is also an important factor. It should be able to cope with hardware and operating system enhancements as well as offering some level of portability between systems with the same operating system. The more flexible a DBMS, the higher the score. Factors such as the number of different machines which could act as host and how much of the system must be changed in order to transfer it will be strongly considered. A

DBMS which runs under the UNIX operating system is considered to be a highly portable system and will receive the highest score.

5.2.1.5. Data Dictionary

A data dictionary (DD) generally serves two purposes within a DBMS. The first is the collection and dissemination of data about the structure of the database, as well as the data itself. Secondly, it serves as a tool by which database standards can be imposed and enforced. Such areas include data element names and usage, coding conventions, security, and database monitoring and statistics gathering. (REF 2, p 6)

The closer a DBMS is tied to its data dictionary, the higher the score. If the DD is a strongly integrated part of the system and its operation, the highest score will be assigned. If the DD is an optional or ancillary feature, the score will be reduced accordingly because the DBMS itself could be used to construct and maintain a DD through custom applications programs.

5.2.2. Important Operational Features/Considerations (Section II)

5.2.2.1. Application Languages

The programming interface which the DBMS supports represents an important aspect of implementation. Researchers have classified these interfaces into two broad categories. The self contained DBMS is characterized by a special language facility (data manipulation language or

DML) but a host language DBMS is based upon a high order language (HOL) and employs its' "call" facility to support the DBMS commands. By using the DBMS' unique DML, applications programs become dependent on the facilities of a particular DBMS. While this may be an advantage for the vendor, it inhibits the flexibility of the user. In general, the commercial DBMSs which provide for the host language capability have proven much more successful than those with only a self contained capability. (REF 2, p15)

The ability of a DBMS to support applications languages and the number supported will be evaluated according to the following criteria. One point will be assigned for each HOL a DBMS will interface with, and one additional point if it is currently supported within AFIT. Five points will be deducted if the DBMS does not have a HOL interface and employs a "unique" interface to the database.

5.2.2.2. Privacy and Security

Privacy refers to the protection of a database from unauthorized use, and security refers to the protection from both overt and inadvertent destruction. The implementation of these concepts often operates at various levels ranging from protection of the full database to individual, single files only or it may cover various combinations of structure data for logical organization. Some systems go to such extremes as restricting the user from any access to the structure data and others allow various combinations depending on special authorization levels. (REF 7, p 3-2)

The security and privacy features available on the DBMS will be investigated by applying the following criteria. Two points will be assigned for each of the following four areas. 1) Does the DBMS provide/control access to the database itself (password, person-id)? 2) Does the DBMS provide/control access to files, records and/or attributes? 3) Does the DBMS provide/control access to the logical view of the data? 4) If passwords are used, are they system generated and can the user change them? Points will be subtracted if the security features rely explicitly on the operating system and if the data base itself can be destroyed or damaged from within or without the operating system.

5.2.2.3. Backup and Recovery

It is evident that we cannot rely on the indefinite preservation of the data in a database. Data in the machines registers or solid state memory cannot be presumed to survive a power outage, for example. Further, no data is completely safe from being obliterated by system software errors. For these reasons, it is essential that backup copies of the database be made periodically, at least once a day if possible. The copy, on a tape or disk, should be removed from the vicinity of the computer and stored in a safe place.

When making a copy, it is important that the copied data represent a consistent state. (REF 17, p 352-353) Recovery functions depend on two basic steps. The first is reconstruction of the data, and the second, reestablishing the relationships among the various records of the database so as to include all modifications up to the failure point. (REF 7, p 3-2)

Of primary concern is the safety of the database to failure and how reliable will it be once restored after a DBMS or system software error.

The thoroughness and effectiveness of the backup and recovery facilities of the DBMS will be examined by the following procedure. One point will be assigned for each of the following criteria. 1) Are the recovery procedures automatic (2 pts) versus user initiated (1 pt)? 2) Is the database safe from being damaged beyond beyond repair? (subtract one point if it makes a difference if the database was in use at the time) 3) Does the vendor offer any guarantees or assistance in the event of an accident? 4) does the DBMS offer facilities for backup (either automatic or manual) instead of using other methods available from the host computer system? 5) Does the DBMS maintain transaction logging and are the files available to the general user (Before transaction logging = 1 pt and After transaction logging = 1)? 6) Can the transaction logging be turned on and off (subtract one point)? 7) Is the logging done without adversely affect the response and efficiency of the DBMS software?

5.2.2.4. Structure Definition

Each DBMS must provide a means by which the DBA describes the individual fields, their format and the conditions for access and any logical relationships. This mechanism is usually called a database description language (DDL) and is often a stand-alone facility and it is via this that the construction of the database, as the user "sees" it, is achieved. (REF 7, p 3-6) These programs can either build tables that are accessed by the DBMS at run time or generate code modules that are entered by the DBMS at run time; the choice is made by the DBA. (REF 2, p 23)

The DBMS software often gives the user many options which may be incorporated at the time of [database] installation. Such items as the number of files, file names, data space sizes, types of access methods, physical file organization must be considered. All these factors must come together to install a database into an operating system on a host computer. (REF 7, p 3-11)

How easy it is for the DBA to define and install a database will be evaluated with the following procedure. Two points will be assigned for each of three main areas. They are: 1) Can the database definition be accomplished using macro procedures or file inputs? 2) Once established, can the structure be used/queried by authorized users? 3) Is the process of defining a database simple and automatic or is it a difficult manual task?

5.2.2.5. Modifiability of the Structure

In a changing environment the database often must be expanded or changed in order to meet new user requirements. Once the database has been installed and loaded with data, changing the structure, adding files or changing their description, can be a potentially dangerous major effort. Large amounts of data must be moved or changed and some may be lost or left irrepairably damaged if the modification is not done correctly and precisely.

The ease and simplicity with which the DBA can change the structure of a database, once installed and loaded, will be examined by the following criteria. Two points will be assigned for each of the following areas: 1) Can attributes be added/deleted without undue impact on the remainder of the database? 2) Can files be added/deleted without undue impact on the remainder of the database? 3) Can logical views be easily modified? 4) Can the database be modified interactively without rendering the database unusable during the process?

6.2.2.5.

Because the database utilizes a file system within an operating system, certain functions must be performed periodically to guard against wasting large amounts of space due to deletion of data. Such wasted space can have adverse effects on the efficiency and response time of not only the database itself but also on the host computer. Some measures must be taken to insure that the database is "healthy" and is as efficient as possible. Normally these measures are transparent to the DBA, handled automatically by the DBMS and no thought is given to their operation. Unfortunately some DBMSs do not incorporate such features and the DBA must rely on certain operating system facilities to specifically adjust the files and remove wasted record areas.

Whether a DBMS supplies automatic compensations for the addition and deletion of data will be evaluated using the following procedure. Two points will be assigned for each of the following three criteria: 1) Must the database files be reorganized periodically? 2) Are these procedures automatic? 3) Can the DBA make the database more efficient? (tunability)

6.2.2.6. User Friendliness & Facilities

The user/system interface (USI) is a critical element of a DBMS, for without it the system will inadequately serve its users. Without the appropriate interfaces for various levels of users the database will fall into disuse. Among these important factors are two which are vital to the users. They are language and interactive capabilities.

The utilization of natural language for communicating with the system will enhance the ability of users to formulate requests and problem definitions. An interactive capability will make it possible to compliment the human thought process and aid management in its decision making activities. (REF 2, p 7)

The friendliness, ease of use and non-procedurality of a DBMS will be investigated using the following criteria. One point will be assigned for each of the following areas: 1) Is the DBMS non-procedural and oriented toward non-programmers? 2) Does it use an english language approach? 3) Can the users get documentation or help on-line? 4) Is the use of the DBMS independent of the users' knowledge of the underlying structure of the database? 5) Does it offer any interfaces to text or file processing facilities? 6) Is it generally user friendly with regards to errors and mistakes? 7) Can queries or other operations be "canned" or saved for future use? 8) Can functions be used within the query language itself and are there any supplied by the vendor?

6.2.2.7. Resources (optional or required)

The nature and complexity of a DBMS occasionally, depending on the hardware and particular implementation, will require special modifications or specific hardware in order to run correctly. On the other hand the features of a given DBMS will often allow it to support some special hardware such as graphics terminals or plotters.

Two points will be subtracted for each piece of special hardware required by the DBMS. Two points will be assigned for each piece of special hardware which the DBMS can support without modification of the software.

6.2.2.8. Update/Retrieval Protocol

Ideally, a database system should permit unrestricted access of all users/programs to any data that they are authorized to use. This presents no difficulties when several users/programs attempt to retrieve data from the same file; the complications arise when more than one wishes to change the database. (REF 16, p 2-27) Intuitively, two process that read and change the value of the same object must not be allowed to run concurrently.

Actually, transactions, when executed concurrently, give rise to dangerous situations such as live-lock, dead-lock and nonserializability. (REF 17, p 324,331) Each of these is a product of the DBMS trying to prevent concurrent operations, resulting in one or more users/programs being temporarily prevented from accessing the required data. The methods by which these are handled by various DBMSs is by no means consistent, nor in some cases, desirable. Whether a particular DBMS will adequately support concurrent, multiple queries and updates, and how well it does it, is a vital question in an environment where many users will access the database daily.

The following criteria will determine which DBMS adequately support multiple query and update functions. Two points will be assigned for each of the following criteria which the DBMS can fulfill. 1) Can the DBMS support multiple query and single update? 2) Can the DBMS support multiple query and multiple update? 3) Is the DBMS multi-threaded? 4) Once a process is locked out of a file, will the DBMS automatically reinitiate the action before returning with an error?

6.2.2.9. Report Generator

The availability and use of a report generator can drastically shorten development time for most applications. It can provide assistance to users in placing the output from a database into the most humanly comprehensible form. Raw data selected from the database can be sorted and/or broken down as the user requires and the format, headings or order easily changed as desired. However, not all commercial DBMSs are packaged with such a facility. In these cases it is necessary that the DBMS provide an adequate interface to a separate, external report generator package.

Three points will be assigned for the first two general criteria. and two points for the third general criteria. 1) Does the DBMS have a report generator which is integrated into the system? 2) Is it an easy to use, non-procedural facility? 3) Does it support the use of macros? Any limiting factors which are discovered will have appropriate points subtracted.

6.2.2.10. Vendor Support

While not essential to the actual operation of a DBMS, vendor support can ease the burden and problems associated with transitioning to a new software system. Many hours of research and study can be eliminated if a vendor provides adequate training and assistance during and after the implementation of the software and subsequent databases. Additionally, just as with any other product, the "service after the sale" will provide much needed help in situations such as hardware changes and/or enhancements and repairs to both the operating system and DBMS software.

The following criteria will determine how much assistance a vendor is willing to provide those who have acquired his DBMS. Points will be assigned as follows: 1) Will or does the vendor provide training? (2 points) 2) Is the vendor available at any time to answer questions or solve problems? (2 points) 3) Is there an established users group? (2 points) 4) Will the vendor repair or modify the DBMS software as required? (1 point) 5) Will the vendor provide updates to the DBMS as new features are developed? (1 point) One point will be subtracted for each item for which the vendor requires an additional charge over and above the cost the the DBMS.

7.2.2.10.

Every software vendor has some kind of documentation available for his product. However, the quality and usefulness of the work plus the price and availability will vary greatly. In an environment such as AFIT where there will be many users of varied backgrounds, each performing unique tasks, the existence of adequate documentation is essential. It would relieve a great deal of tension and frustration from the users if problems can be solved and questions answered quickly and with a minimum of trouble.

The following criteria will determine whether or not the vendor of a particular DBMS can provide AFIT with adequate documentation of his product. Points will be assigned for each of the three following criteria as follows: 1) Is the documentation clear, readable and easy to use? (3 points) 2) Does it answer questions which the majority of average users would be concerned with? (2 points) 3) Is any of the documentation available on-line? (3 points) Two points will be deducted if the vendor charges extra for documentation or for more than a few copies.

7.2.3. Desirable Features (Section III)

7.2.3.1. Batch/On-line Operation

The flexibility of a DBMS is partially determined by the methods by which users can access the data. If users are restricted to only a single mode of operation, delays can be incurred by requests which require either long or very short periods of time in which to be accomplished. Ideally a DBMS should be flexible enough to allow a user the option of performing his tasks while on-line and waiting for the results or asking the DBMS to perform the job in a remote or batch environment.

Two points will be be awarded for any DBMS which meets the following criteria: 1) Is the DBMS oriented toward on-line operation? 2) Can the DBMS support batch operations? 3) Can the DBMS support both batch and on-line operations concurrently?

7.2.3.2. Utilities/Functions

The ability to perform complex or redundant procedures in a quick and easy manner can be an invaluable aid to the users of a DBMS. Utilities which allow certain mathematical and/or logical operations on the data as well as those which ease textual manipulation will increase the DBMSs usefulness to the user as well as increasing his productivity. For example, it is often desirable to initialize a database with data from other existing files. A facility for automatically converting a non-database file into a form which is readily moved into the database would be highly useful. (REF 7, p 3.1.5)

The kinds and type of utilities are not of interest as much as the ability of the DBMS to use them. If a DBMS easily supports utilities and functions, users may freely create as many as they wish to satisfy their needs. One point will be assigned to a DBMS which fulfills each of the following criteria: 1) Does the DBMS provide for utilities and functions? 2) Can the user create his own utilities and functions? 3) Are these utilities permanently available until removed by the creator? 4) Can these utilities be shared/used by others? 5) Does the vendor provide a set of utilities for the DBMS and its operation?

7.2.3.3. Accounting Facilities

Certain users of the proposed AFIT database have expressed a desire to perform accounting activities in conjunction with their normal database activities. If available, any DBMS which incorporates such a facility will be awarded six points.

7.2.3.4. Field Value Enforcement (data integrity)

There are generally two types of problems associated with the maintenance of data integrity. First is the problem of the correct type of data for a specified field, e.g. alphanumeric, binary, etc. The second relates to the value of the data to be placed in the fields. For example, there may be automatic facilities in the DBMS which specifies that a data value may not exceed a certain range. (REF 7, p 3-2)

The ability of the DBMS to assist in maintaining data integrity will be investigated by the following criteria. One point will be assigned to a DBMS which fulfills each of the following criteria: 1) Will/does the DBMS enforce field values? 2) Will/does the DBMS enforce data types? 3) Is this defined/changed by the DBA instead of the user? 4) Does the DBMS notify the user clearly and accurately as to what has happened? 5) Can the parameters be examined by the user?

7.2.3.5. Key Value Enforcement

Each DBMS which enforces the uniqueness of predefined key fields will be assigned six points. If there is a limit to the number of key fileds which may be defined in a given record, one or two points will be deducted depending upon the amount.

7.2.3.6. Secondary Indexing

If a particular DBMS allows for secondary indices to be placed upon the attributes within a record, three points will be assigned. Ability to fully invert the structure will merit another three points.

7.2.3.7. User Usage Statistics

The DBA or other authorized levels of management often are concerned with the number and frequency of accesses to the database. Areas such as time and type of query, which users (by department) access the database and what files they use are commonly of interest. If the DBMS has facilities for maintaining these types of statistics, six points will be awarded.

7.3. RESULTS and CONCLUSIONS

7.3.1. General Procedure

As much information about available DBMSs was gathered from a wide variety of sources as was possible. The most valuable was an article which appeared in a September 1981 issue of DATAMATION (REF 11) which presented a short synopsis of each of DBMS on the market. With this as a guide, further information was gathered as required from other sources, such as DATAPRO and other articles in periodicals. Several weeks were devoted, in whole or part, to studying this data and talking to several vendors in order to gain a thorough understanding of the available DBMSs and what the current state-of-the-art is.

The evaluation was carried out by searching for answers to the set of questions in the evaluation section (Section 3.2.1) and assigning points according to the pre-defined rules. When there was no data or the data was insufficient so that a firm determination could be made concerning a item, a value of zero was assigned. In some cases the vendor was contacted for clarification and additional detailed information on those DBMSs which scored a 70 or higher (the top 14 systems) on the evaluation was requested. The total values were calculated for each DBMS and they ranked accordingly (Appendix IIB). A chart showing how each DBMS scored in each category may be found in Appendix IIC.

7.3.2. General Comments on the Procedure

Across the board there is little specific, detailed information available to the public on DBMSs. There are several sources from which very general information may be obtained, but little of the type required by this evaluation. DATAPRO has a fairly detailed breakdown on DBMSs (including most of what was needed for those systems they listed), but mainly on those systems oriented toward IBM and other larger computer systems. It appears that a large segment of the market has been overlooked and DATAPRO's descriptions exclude many "recent" and very capable systems. Sometimes that information which was available was ambiguous or misleading as to the actual capabilities and functions of the DBMS.

Another aspect which was difficult to pin down were the DBMSs which were available from within the Federal Government. The General Services Administration (GSA) Federal Software Exchange program either did not contain the desired information or simply did not list possible systems. One DBMS (RIM) which was written under a government contract, known by the author and others at AFIT, was evaluated but did not appear on the list. This most likely is a failure on the part of the organizations to list their systems and not on the system itself.

The evaluation as discussed in Section 3.2.1 generally turned out to be too detailed for the level of information received.. Even though the items to be evaluated are crucial to the smooth and efficient operation of a DBMS, vendors usually do not widely publish sufficient details on the operation of their product. In those instances where a vendor was contacted and asked specific questions (from the evaluation sheet), they readily supplied the information. Because of this situation, this evaluation could only be 100% fair and complete if each vendor is contacted individually and asked the set of questions. As it turned out, however, those systems which were rated low because of a lack of information probably could not been considered as likely candidates because the type of computer which could support them was limited or other significant features, for which there was sufficient data, could not fulfill AFITs needs anyway.

In spite of the lack of data in some areas and because of the way in which the evaluation was structured it is felt that those systems which scored the higher points are the most suitable for AFIT. It is felt, very strongly, that the evaluation was sufficiently encompassing and well rounded to adequately reflect the needs of AFIT. Additionally, as was hoped, the best possible DBMSs were identified through a fair and impartial process.

7.3.3. Conclusions

Several important conclusions were reached during the course of this evaluation. They fall into two general categories. One deals with AFIT itself and the other concerns the DBMS market.

Most important is the fact that the capabilities which AFIT requires of a DBMS are well within the current state-of-the-art in DBMS technology. The systems now being marketed are extremely sophisticated and written mainly for the non-programmer. They provide a great deal of flexibility for the user and the organization alike in both the manipulation of the data and the database structure itself. Additionally, they virtually all but eliminate the need for applications programs in organizations which have information requirements similar to AFIT. Extremely sophisticated, english like query facilities combined with good security measures, report generators and accounting capabilities provide a DBMS which can be virtually tailored to the individual users.

Two less significant facts were also discovered. As far as industry views databases and their uses, AFIT represents a rather small application. This is in spite of the potentially large amount of data which will be in the database. Database applications are generally classified by the amount and frequency of queries levied against the data. Even at peak usage, AFIT will not amount to a significant level, as defined by industry. Another thing learned was that AFIT has a rather unusual mixture of computer hardware on which to implement a DBMS. There are a limited number of DBMSs currently available which can run under the UNIX operating systems. However, those which are available are very powerful and popular. As few as these were, the number of DBMSs capable of running on the HARRIS minicomputer are fewer still. This situation severely limits the range of possible DBMS candidates for AFIT.

Because of the large number of DBMSs which were evaluated, some general observations may be made about the market as a whole. Primarily, there are many good DBMSs which will more than satisfy AFIT's requirements, and at reasonable price.

Secondly, the majority of the DBMS market appears to be oriented toward IBM and DEC hardware. This is not all to surprising because these two vendors manufacture the majority of hardware presently in use throughout the world. On the other hand, there does seem to be a trend by smaller software companies to fill the gap and provide capable and reliable DBMSs for a larger variety of hardware and operating systems. It is probably safe to state that there is at least one DBMS which will operate on almost every computer currently commercially available. This does not necessarily include microcomputers even though they too are being quickly accounted for.

7.3.4. Recommendations

Recommendations for the DBMS to be utilized within AFIT to support CADIS are based upon the current hardware configurations at AFIT, the information requirements and performance characteristics expressed in the interviews as well as the current state of DBMS technology.

It was determined at the conclusion of the evaluation procedure that those system which scored a 70 or higher were to be considered as prime candidates for implementation. These systems are the top 14 systems out of 44 evaluated. Cost figures are not included in the recommendations nor were they considered heavily because a specific price must be negotiated with the vendor and might very well be lower or higher than that quoted in the literature, depending on the individual situation.

No single specific recommendation for the DBMS or computer system has been given. What has been done, however, is a list of the best alternatives, as viewed by the authors, has been drawn up from which the final choice might be made. This will allow the alternative which best suites AFIT at the time to be used. The following are recommended as possible solutions for the implementation of CADIS:

(1) Acquire the ORACLE DBMS, currently on the GSA price schedule, and implement it on the PDP 11/34, PDP 11/60 or VAX 11/780 under the UNIX operating system.

(2) Acquire the SEED DBMS, currently on the GSA price schedule, and either remove UNIX from one of the PDPs and reinstall the original operating system (SEED does not yet run under UNIX) or acquire another DEC computer system or some other which will run SEED.

(3) Acquire SEED and negotiate with the vendor to make it run on the HARRIS mini computer. The vendor has stated that this is possible, and they are considering it now, but the actual cost, if any, would have to be negotiated.

(4) Acquire a complete TOTAL DBMS package and install it on the HARRIS. The package available from AFIT/EN is not complete and would not fulfill the needs of the users because those modules which make it so, like the query facility and data dictionary, are separately priced items and must be acquired as such.

(5) Acquire, or reach a usage agreement with a WPAFB organization for the use of, a PRIME computer system and install/use the INFO DBMS.

(6) Acquire the latest version of the RIM DBMS, currently available from BOEING Computer Co. under a NASA contract, and install it on the VAX 11/780 in AFIT/EN. Under this option, communications would have to be provided.

(7) Acquire an IBM System 38 Database Computer.

The two highest ranking DBMSs, ORACLE and SEED, are almost equally qualified for implementation of CADIS, and should be used. The capabilities and functions of each are almost identical in all respects. Each has an advanced level of data security, a data dictionary, fully capable report generators, database and data maintenance facilities and a highly sophisticated and useful CRT screen formating facility for applications development.

Two of the most significant features of these DBMSs are the security and CRT screen formator. Security is controlled not only with passwords and person-ids to the attribute level at least, and one (ORACLE) allows the logical data views to specify restrictions on a subset of data within a relation, such as STUDENTS. This means that each directorate can control the privileges on their data, not only for themselves but for the rest of AFIT as well. As far as can be determined, these are the only DBMSs currently available which offer such extensive and flexible security measures (others merely allow password and/or person-id protection).

Almost as significant as the strong security is the existence of an easy to use (user oriented) and very capable CRT screen formator. This feature could all but eliminate the need for applications programs at AFIT. With the use of any CRT which allows for direct cursor addressing, a user may designate protected and unprotected areas for data entry, specify fields which cannot be left empty or blank, specify special edit checks on the data (this includes having the DBMS check the database to see if a value is already present in the database; especially useful for codes). In addition, the prompts and error messages the user sees can be specified as well as special on-line, application dependent, help files.

IV APPLICATIONS PROGRAMS DESIGN

8. GENERAL DESIGN STANDARDS

8.1.1. Goals and Desires

"We build systems like the Wright brothers built airplanes - build the whole thing, push it off a cliff, let it crash, and start over again." (REF 19 p xi)

The goal of this chapter is to present a detailed top-down analysis and design of a reliable set of applications programs which will access the CADIS database and fully satisfy the present and future operational requirements of AFIT/CI. It is intended that the set of documentation provided by this chapter will make it easier for those who will actually write the programs to understand the problems of AFIT/CI as clearly as the authors presently do and allow them to translate that understanding into a usable, maintainable set of programs.

By relying on the techniques of software engineering, it is more certain that the actual problems, both general and specific, will be understood by the designer, users and implementors before any code is ever attempted. This will insure that all the operational needs are met and the users will have a system which works, works correctly, and is easy to use. The single most significant advantage of using these techniques is that the system will be easier to implement, code and maintain at a much reduced cost.

"In the 1950's and 1960's, machine costs associated with systems development exceeded people costs...[however,] in todays environment people costs far exceed machine costs...". (REF 18 p 29) It is because of this statement that the focus of this section is to be upon the functional description and design of the programs and not an actual implementation. Because a decision has yet to be made as to which DBMS, which hardware and what Higher Order programming Language (HOL) will be used to implement this project, it is very imporrant that a well thought out design be agreed upon early. It is important to note that "an important aspect of structured design includes designing on a logical level before continuing to the specifics of a particular physical solution. A logical design is easier to communicate because it is more concise, less technical and uses graphics [aids]. Once the users have agreed to the logical requirements of the system, [it] is transferred...into the modules of the system." (REF 18 p 31)

To summarize, our goal is to provide a top-down, structured design of the applications programs required by AFIT/CI. By so doing, it is assured that they will meet the operational specifications and will be easy to implement and maintain.

AFIT/CI requirements dictated that there be two separate software packages designed. While the general design for both systems was worked on by each author, the specifics and details were basically divided along the AFIT/CI organizational lines. Capt. Ricks was responsible for the Financial System (CIV) and Lt. Colburn handled the Personnel System (CIM/R).

8.1.2. Procedure

The general procedure to be followed is one which combines aspects of almost all of the procedural and graphic techniques currently available, but which has been developed and extensively used by Capt. Ricks. The only assumption which must be made for this technique to be successful, is that the language used to implements it must support a "call" mechanism.

This procedure centers around a hierarchy chart and two written documents. The chart is a modified HIPO chart (REF 19 p 46) which incorporates elements of Structure Charts (REF 18 p 134) as well as Structured Analysis and Design Techniques (SADT) charts. The two documents used are a functional Description (FD) of the software and detailed, module by module descriptions of the components shown in the hierarchy chart.

Development of the hierarchy chart is accomplished in two distinct stages. First the major functions or operations which are performed are identified and listed in the logical order of performance. Next, these functions are refined by incorporating the screen displays, commands and other operational features of the programs as specified by the user. Once the sequence of events and software interaction have been agreed upon by the designer and the user, the chart is modified to reflect the detailed functions, and if desired, the control logic which the applications programmer will actually use during implementation.

The screen displays are then finalized and collected together, along with appropriate inputs, outputs and user interactions. This is documented and becomes essentially a pictorial of just how the system will act and look when the user gets on the terminal. This narrative is the Functional Description (FD) of the software and will become a users manual once the system is completed. It is against this document that the programmer must verify his formats and responses during the coding and testing of the system.

The third element, and perhaps the most important, is the detailed description of each function represented on the hierarchy chart. When the functions have been broken down sufficiently, according to structured analysis principles (REF 18 & 19), they will in essence represent the actual modules or subprograms which must be coded.

The level of documentation includes the inputs and outputs of each module as well as a structured english (REF 18 p 321) description of the procedures needed to transform the specified input to the desired output. By doing this for each module, the programmer can transform this set of specifications directly into the desired HOL statements required for implementation.

It is assumed and encouraged that the tenets of structured programming (coding) be followed strictly and zealously to insure that the design is accurately reflected in the actual code.

Each of the above documents are listed in appendix III. The hierarchy charts are appendix IIIB & IIIE, the Functional Descriptions are appendix IIIC & IIIF and the modules descriptions are appendix IIID & IIIG. Also listed in appendix III are the new relations for the database which were developed after further, in depth, analysis of the AFIT/CI operations. These relations must be added, or modified in some cases, in order for the software to produce the desired results.

The above procedure for the design of applications software is not significantly different, in principle, from those advocated by current software engineering literature and practices. What has been done however, is to combine those aspects and details which work best and ignore those which the author feels constrain or limit the productiveness of software designers. This process has been employed by the author on many systems at several locations and under differing situations with very good success.

8.1.3. Documentation

A specific format has been adopted to document the modules within the hierarchy charts. Each module is listed by number and a descriptive name, along with what it does (function) and its input and output. A variable is considered an input or output based upon its value and its use within the module. For example, a parameter whose value is set or reset within the module is considered an output while one which is passed in for information or control is an input.

Probably the most important portion of the documentation is the description of the procedure or workings of the modules. A highly structured english (REF 18 & 19) description is used to show which and where each parameter is set, what subroutines are called, and the governing logic at each level.

In several modules the reference to "get input" and "get line" is used. "Get input" is used whenever a single value is to be entered (e.g. from a menu) and "get line" is used when an entire line of values (e.g. a line from an update display) is entered. A basic assumption is that when the line is provided to the module all spurious characters are stripped away and the only thing left are the input fields with possibly a delimiter character between them. The actual mechanics must be left to the programmers and the specific language used.

Another function which is referred to in general terms is that of "edit input". Each field that is changed by the user must be checked for validity and correctness of format. These details were also left for the programmers because it was not known exactly what all the criteria for each field were. Any error checking should notify the user immediately (at the bottom of the screen) and allow him to correct his mistakes. Of primary consideration is the line numbers on the update programs. The easiest way of displaying these lines is to build them and place them in an array within the program. When they are displayed, the line number corresponds to their location within the array. If the line numbers are changed or messed up in any way, the program will not be able to guarantee the accuracy of the results.

In some of the modules a variable called "ifun" is used. This variable represents the value of the function or command (FUN/CMD) column on the update displays as input by the user. If need be its value will be assigned to the master control parameter FUNCTION.

8.1.4. Software Requirements Analysis

The software requirements of AFIT/CI were gathered by means of a simple questionnaire (Appendix IIIA). It was developed jointly by the authors and given to the data automation representative of AFIT/CI to complete. Responses to this form, in various forms and formats, provided the basic groundwork for both the data requirements and the performance specifications and criteria for the software. This initiated a continuing process of question and answer sessions with the users which culminated in the software design presented here.

8.1.5. Design Approach

The approach to the design of the AFIT/CIV software was as follows:

(1) Interview the program manager representatives from AFIT/CI.

(2) Formulate the initial design requirements based on interviews and the questionnaire (Appendix IIIA).

(3) Develop the initial, high level, top-down interface design.

(4) Review the initial design with the program manager representatives.

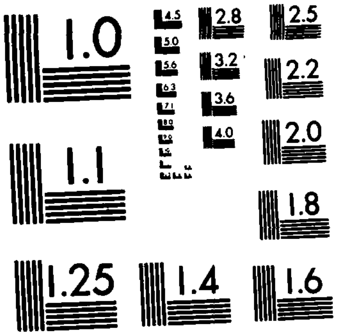
(5) Re-evaluate the design and make appropriate changes requested by the user.

(6) Consolidate all notes and rough drafts and prepare the results for final presentation and documentation.

8.1.6. Required Hardware

In addition to the central computer system which uses the DBMS, the only additional hardware required to use the software, as presented here, are terminals. The programs have been designed to use the facilities of a CRT which uses direct cursor addressing and has the ability to produce protected and unprotected fields. All displays and programs operate on this assumption and were designed to be used by any user simply and easily. This will simplify data entry and manipulation, reduce wasted time due to errors and confusion, and provide a system which is more readily accepted by the user.

AD-A124 647 ANALYSIS OF INFORMATION REQUIREMENTS AND DESIGN OF THE 2/4
CONSOLIDATED AFIT D..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..
UNCLASSIFIED J S RICKS ET AL. DEC 82 AFIT/GCS/EE/82D-11 F/G 5/2 NL


1.0
4.5
5.0
5.6
6.3
7.1
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

8.2. AFIT/CIM/R SOFTWARE DESIGN

(This software design was authored by Lt. Robert Colburn)

Todays Air Force requires a seemingly ever increasing supply of well educated doctors, engineers and scientists. Much of the responsibility for fulfilling these needs fall upon the AFIT Civilian Institution Programs directorate.

AFIT/CI is responsible for managing the civilian institution academic programs, medical training, Education With Industry (EWI) and the MinuteMan Education Program (MMEP). These programs, plus many others, and their students are all managed by either CIR (regular programs) or CIM (medical programs).

Approximately 5000 widely dispersed students (geographically), in approximately 300 universities and institutions are managed by AFIT/CI. This is accomplished by 15 program manager teams directly interfacing with the students, the civilian institution in which they are enrolled and the Air Force, all on a daily basis. Presently, all record keeping and data retrieval is carried out using file folders and manual methods. This inefficient and inadequate mode of operation is severly handicapping the ability of the program managers to provide the required guidance to their students and is overloading the current manpower level. AFIT/CI has recognized this fact for a long time as being unsatisfactory and detrimental in terms of meeting daily operational requirements. Given the increased emphasis on engineers and scientists within the Air Force and the corresponding increase in enrollments in civilian institutions to meet this goal, the current situation will only become worse.

In order to properly cope with the present and future operational overload, AFIT/CIM/R has chosen to automate its information processing system. It is within the context of this background and a desire to solve and enhance operational capability, that the design of a special set of applications software has been undertaken.

It is a recognized fact that AFIT/CIM/R has an information problem that is impacting their ability to handle the civilian institution programs. The underlying assumption of this thesis is that a set of user friendly applications programs specifically tailored to meet their needs and acting as an interface to the CADIS database will alleviate both current and projected workload and information problems.

While this software will go a long way toward helping AFIT/CIM/R, such a system is not a panacea for all their problems. However, in terms of efficiently storing, organizing and quickly processing their large amounts of data, this approach has repeatedly proven effective throughout the commercial, federal and academic arenas.

8.2.1. General User Requirements

Interviews with program manager representatives from AFIT/CIM/R indicated that there are four primary areas which they use frequently and urgently need automated. These areas are:

(1) Student Biographical Data

(2) Education Plan data

(3) Memorandums For The Record

(4) Suspense Dates and Actions

The relations to support these areas are already present in the CADIS database with the exception of the suspense data. This may be found in appendix IIIH. Because the data management needs of the two divisions , CIM and CIR, are so similar, only one logical data view has been provided for both.

8.3. AFIT/CIV SOFTWARE DESIGN

(This software design was authored by Capt. Jeffrey Ricks)

The Accounting/Data Control Division of the Civilian Institution Programs Directorate (AFIT/CIV) is responsible for maintaining all the financial matters pertaining to the thousands of geographically dispersed students enrolled in civilian universities for the purpose of obtaining an AFIT sponsored degree. AFIT/CIV currently handles financial data and is responsible (fiscally) for approximately 5000 students enrolled in some 300 universities, in various educational programs, across the country. In the performance of their duties, they have no subordinate level of administration and must be in contact with each student at each university throughout the year. Currently this task is being accomplished with manual filing systems and approximately eight office personnel.

The major duties which must be accounted for include:

(1) Payment of tuition and fees for each student. Such payments may be made either directly to the student or to the university, depending upon the school and specific program, up to four times per year.

(2) Reimbursements for approximately 1500 students for authorized expenses incurred; up to four times per year.

(3) Preparation and continual maintenance of individual financial records on tuition and all fees for approximately 3000 students and personal reimbursements for approximately 1500 students.

(4) Generation of routine statistical and audit data on students, payments and institutions for AFIT plus special requests from HQ/USAF, AFMPC and offices throughout DOD.

(5) Tuition and fee charges for approximately 300 universities. This translates into a wide variety of tuition rates (resident vs nonresident), fee charges (lab, parking, etc), academic term breakdowns (quarter, semester, etc) and billing procedures.

Maintaining financial data plus the tracking of both students and universities, spread all across the country, renders the verification of charges and the posting and payment of bills by manual means virtually impossible. Additionally, no automated means for the payment of invoices and reimbursements, bookkeeping, as well as performing cost tracking and forecasting, budget analysis, statistical analysis and man year load analysis currently exists within AFIT. The result is that the current workload has already surpassed the authorized manpower and the quality control and audit capabilities for a budget which exceeds 4.5M annually is severely threatened. Additionally, AFIT/CIV is frequently several weeks or sometimes months behind in the prompt payment of bills.

Among the major problems which currently exist are no forecasting capability of any type in any area, inability to maintain historical student and institution data, no direct (automated) access to files and information, inability to use and maintain over 75% of the data items essential for financial program management, no timely reporting in the required formats, and no automatic verification of data.

8.3.1. General User Requirements

The AFIT/CIV directorate must have the ability to automatically update and review the invoices and vouchers for each institution. They must be able to record the amount due, the date the invoice was received and match it with a specific AF voucher for payment of the amount due. Additionally, they also must be able to cross reference this invoice to the students for which it covers.

Each student must have a record kept which shows a detailed breakdown of the expenses he has incurred, the amount, the date and when they were paid (cross reference to the school invoices & vouchers). The general biographical data will be handled through the program managers from AFIT/CIM. Additionally, there must be a means of generating a payment list which can be sent to the local finance office notifying them to issue checks for the listed students for the specified amount.

AT each school there are individual contacts with which the office personnel routinely deal. An automated means of listing and updating each POC, their current phone number and address must be provided.

In order to make the general programs work correctly, the database must keep track of the costs of all the various programs, schools and their facets. Each school has different charges for tuition, both graduate and undergraduate, as well as the many fees which may be charged. By combining the current student and school data with these figures, the required cost forecasting will be able to be accomplished.

BIBLIOGRAPHY

(1) "A Buyers Guide to Data Base Management Systems", DATAPR , 70,3: Section 70e-010-61, 1982.

(2) Aeurbach, "Guide to Data Base Management", Auerbach Publish s Inc, Philadelphia PA, 1975.

(3) AFIT ADP Master Plan, AFIT Directorate of Data Automation, FIT/ACD Wright-Patterson AFB, OH: 1982.

(4) Allred, D. S., Consolidated AFIT Database, MS Thesis, Wright Patterson AFB, OH: School of Engineering, Air Force stitute of Technology, Dec 1980 (AFIT/GCS/EE/80D-3).

(5) BIS Applied Systems Ltd., Data Base Techniques, QED Informa on Sciences Inc, Wellesley MA, 1980.

(6) Cardenas, A. F., Data Base Managements Systems, Allyn & Bac , Boston MA, 1979.

(7) Cohen, L. J., et. al., Data Base Management Systems, QED In rmation Sciences Inc, Wellesley MA, 1981.

(8) Cohen, L. J., Pre-Data Base Survey, Performance Developmen Corp., Princenton N.J., 1979.

(9) Curtice, R. M., Planning for Data Base Systems, QED Informa on Sciences Inc, Wellesley MA, 1976.

(10) Date, C. J., An Introduction to Database Systems, (Third ition), Addison-Wesley Publishing Co, Reading MA, 1981.

(11) Dieckmann, E. M., "Three Relational DBMS", DATAMATION, Sept 981, PP 137-148.

(12) Flores, I., Data Base Architecture, Van Nostrand Reinhold ., New York, 1981.

(13) Hutt, A. T. F., A Relational Data Base Management System, J n Wiley & Sons, Chichester England, 1979.

(14) Krass, Peter and Hirsch Wiener, "The DBMS Market is Booming A Survey of the Database Systems Market...", DATAMATION, 27 3-171, (Sept 1981).

(15) Martin, J., Computer Data-Base Organization, Prentice-Hal , Inc., Englewood Cliffs, N.J., 1977.

(16) Palmer, I., Data Base Systems: A Practical Reference, QED Information Sciences Inc, Wellesley MA, 1982.

(17) Ullman, J. D., Principles of Database Systems, Computer Science Press Inc, Rockville MD, 1980.

(18) Weinberg, Victor, Structured Analysis, Yourdon Press, New York NY, 1980.

(19) Yourdon, Edward, Techniques of Program Structure and Design, Prentice-Hall Inc., Englewood Cliffs, N.J., 1975.

FROM: AFIT/NR 16 July 1982

SUBJECT: Consolidated AFIT Database

TO: AFIT/CCE DA PA DP RM ACD LD
ED RR EN LS DE CI CAE

1. On 3 July 1982 the Computer Configuration Board (CCB) met to discuss the need for and timeliness of various solutions to the administrative problems continually faced by AFIT. After considering several presentations on both past efforts and current projects, including current thesis projects, the CCB decided that in order to cope with the ever increasing student load and the associated administration thereof, a database management system (DBMS) could support AFIT and should be implemented as a long range project. The intention of such DBMS would be to consolidate all the information required by the departments within AFIT and provide simple and easy access to all authorized personnel.

2. The first step in such a project is to gather all the information which each department requires to perform its duties and derive a database structure which will adequately serve all the intended users. Two AFIT/EN students, Capt Ricks and Lt Colburn, have volunteered to gather the needed information as a part of their thesis investigation. Capt Ricks and Lt Colburn will be gathering various kinds of data, performing analyses and making specific recommendations to the CCB concerning a consolidated AFIT DBMS. Their first task is to contact each of the departments within AFIT in order to determine their information and report requirements and, if possible, the specific data elements they use.

3. It is requested that each addressee designate someone to meet with and provide Capt Ricks and Lt Colburn the data they require so significant progress may be made toward defining our information requirements. They will be contacting each addressee on an individual basis to arrange meetings.

4. If you have any questions, please contact Major Lillie, extension 52024.

SIGNED

L. E. WOLAVER
Dean for Research and
Professional Development

Appendix IA

WORK SHEET

OFFICE SYMBOL: CAPT. RICKS
LT. COLBURN

POINT OF CONTACT: EXT. 52194

PROPOSED INTEGRATED AFIT DATABASE RECORDS AND ATTRIBUTES

(NOTE: EN's Database)

STUDENT DATA: SSSN, Name, Rank, PAFSC, Ed Code, DOR, DOC, DOB, POB, Phone, Duty Phone, Address, Emer Address, Religion, Aero Rating, Losing Command, Military Spouse, Years of Service, Last Organization, Position Title, Duration

BOX NUM: BOX NUM

SECTION: SECTION NUM, Grad Date, Enter LDate, Num in Section, LDRSSN

THESIS DATA: THESIS NUM, Title, Sponsor, Location, Classification

GRADES: SSSN, Course Num, Quarter, Grade

FACULTY DATA: FSSN, Name, Rank, Phone, Home Phone, Room, Office Symbol, Department, Job Title, Begin Date, End Date, Address, DOB, Spouse, SDOB, Marital Status, Num of Dependents, Dependent Names, Service, Religion, Photo Date

AWARDS: FSSN, Award, Award Date

PUBLICATIONS: PUBTITLE, FSSN

TDY: TDYDEST, FSSN, Begin Date, End Date, Tripcost

COLLEGES: FSSN, College, Begin Date, End Date

DEGREES: FSSN, Degree, Degree, College, Year

PUBLICATION DATA: PUBTITLE, Date, Topic, Source, SSSNAUT

COURSE DATA: COURSE NUM, Credhrs, Lechrs, Labhrs, Description, Outline, Sizelmt Sizelmt

TITLE: COURSE NUM, Title

PREREQ: COURSE NUM, Prereq Num

BOOKS: BOOK TITLE, Author, Publisher, Num Avail, Price

BOOK ORDERS: AUTHOR, Book Title, Order Num, Num Ordered

ORDERS: ORDER NUM, Order Date, Due Date

COURSE BOOKS: Course Num, Author, Title

OFFERINGS: Course Num, Quarter

DATES: Quarter, Date

QUARTER: QUARTER

ROOMS: ROOM, Bldg, Num Allowed

SCHEDULE: DATE, Time, Room, Course Num

PRESENTATIONS: FSSN, Date, Presentation Title, Location, Org

COMMITTEES: FSSN, Committee

STUDENT AWARDS: SSSN, Award, Award Date

STUDENT SCHOOL: SSSN, College, Course Num, Grade

STUDENT DEGREES: SSSN, Degree, College, Year

INSTRUCTOR: FSSN, Course Num, Quarter

ATTRIBUTE EXPLANATION

RECORD	ATTRIBUTE	EXPLANATION
SECTION	LDRSSN	Section leader's SSN
THESIS	LOCATION	Sponsor's location DATA
PUBLICATION	SSSNAUT	Student coauthor's SSN DATA
PRESENTATION	ORGANIZATION	Organization the presentation is given to
FACULTY DATA	FSSN	Faculty members SSN
FACULTY DATA	SDOB	Spouse's date of birth
COURSE DATA	SIZELMT	Maximum number of students which are permitted to enroll

WORK SHEET

OFFICE SYMBOL:

POINT OF CONTACT:

VERSION 1

REGISTRAR'S DATA BASE

STUDENT-RECORD: SSAN, CLASS (e.g. 82-D), PROGRAM (e.g. GCS), Student-code (e.g. RS=resident, CI=civilian institution), last-name, first-name, middle-initial, name-ps (e.g. jr., sr.), student-title (e.g. 2LT, MR.), rank-type (e.g. GS,O), rank-level (e.g. 12, 1), date-of-birth, AFIT_school_code (e.g. 1=EN, 2=LS), manning-code (e.g. 1=AF officer, regular), sex, last-majcom, output-majcom, aero-rating (e.g. 1=pilot), ethnic-group (e.g. 1=white), prior-degree (e.g. n4iyy), undergrad-GPA, marital-status (e.g. 1=single), primary-specialty-code, entry-date (begin AFIT), graduation-date, departure-date, degree-granted (e.g. 2=M.S.), prior-AFIT (no. of mths), education-code, folder-location (e.g. 1=RR, 3=CI), departure-reason (e.g. 1=graduated with degree), transfers (no. of times transfered programs), trans-in-date, trans-out-date, grade- point-average, GRE-verbal, GRE-quan, GRE-total, GRE-advanced, GMAT-verb, GMAT-quan, GMAT-total

STUDENT-RECORD FIELDS UNIQUE TO RESIDENT STUDENT RECORDS (RS): resident, status (1=full time), student-number, student-box number, adviser

STUDENT-RECORD FIELDS UNIQUE TO CIVILIAN INSTITUTIONS RECORDS (CI): civil-inst, university, academic standing (e.g. 2=probation), suspense, speciality-code, legal-residence, thesis-dis-approval (e.g. A=approved), medical-accession-source, previous-medical-degree, present-medical-program, medical tours, MEDCAT-exam-scores

STUDENT-RECORD FIELDS UNIQUE TO TRANSFER RECORDS (TR): transfer, old-info (info unique to old record), old-code (previous student code)

WORK SHEET

OFFICE SYMBOL:

POINT OF CONTACT:

VERSION 2

REGISTRAR'S DATA BASE

STUDENT RECORDS SYSTEM

COURSE TERM FILE DATA: course section number, activity type (e.g. LEC=lecture), term (e.g. 821=1982 winter), course section title, section cntrl (special processing, e.g. I=informal credit activity), minimum credit (min. no. of credit hours for a course), maximum credit, variable credit (e.g. 3=credit to be arranged), special grading (e.g. C=credit/no credit only), course level (e.g. G=graduate credit), course-college, course-college & dept, course-number, course section, course status (e.g. O=open), wait list code (e.g. N=no wait list), resection code, room size (e.g. 5=space for over 50), room requirement (e.g. C=classified room), schedule print (e.g. N=do not print on published schedule), schedule notes 1 (e.g. L=taught on an independent study basis), schedule notes 2, schedule notes 3, crs section (e.g. S=special offering), crs start date, crs end date, exam schedule code (e.g. Y=yes, schedule exam), term type (e.g. 1=winter), proj enrollment, enrollment limit, minimum enrollment, enrolled tally, demand tally, drop tally, wait list tally, non-reg tally (students enrolled with out formal regristration), add tally, audit tally, 1st meet days (e.g. M=Monday), 1st meet starts (e.g. 0850=8:50 AM), 1st meet stops, 1st meet bldg (e.g. 640=bldg 640), 1st meet room, 2nd meet days, 2nd meet starts, 2nd meet stops, 2nd meeting bldg, 2nd meeting room, 3rd meet days, 3rd meet starts, 3rd meet stops, 3rd meet bldg, 3rd meet room, 1st inst ssno (1st instructors ssn), 1st inst load (%of course credit 1st inst is credited for), 2nd inst ssno, 2nd inst load, 3rd inst ssno, 3rd inst load, 4th inst ssno, 4th inst load, 5th inst ssno, 5th inst load, 6th inst ssno, 6th inst load, 1st inst name, group contact hours (total no. of hrs per week faculty members meet with students as a group), indiv contact hours (total no. of hrs per week ea. student meets with the faculty member on an individualized basis), dept of record (e.g. EENG=electrical engineering), ctf maint. date (date of last maintenance on ea. record)

STUDENT BIOGRAPHIC AND DEMOGRAPHIC DATA (STUDENT KEY FILE ITEMS): special name flag (indicates special spelling or punctuation of student's name), student identification (student's ssn), re-admit term (indicates term a withdrawn student wishes to return), country, info release flag (e.g.

N=do not release any info), grade mail flag (where grades are to be mailed to, e.g. P=permanent address), partial rec flag (e.g. N=no, record is complete), advisor ID no., academic elig (e.g. Y=yes, automatically register), cumulative earned ho (cum. hrs. earned), cumulative GPA, calc required term, current college (e.g. EN=school of engineering), current classification (e.g. CM= master's candidate), curent degree (degree program student is currently enrolled in, e.g. MS=mast of science), current major (e.g. MATH=mathematics), degree expected term (expected graduation term), deg check out term (expected degree requirements completion term), deg ckout status (e.g. 1=filed graduation plans), administrative hold (e.g. H=records being held)

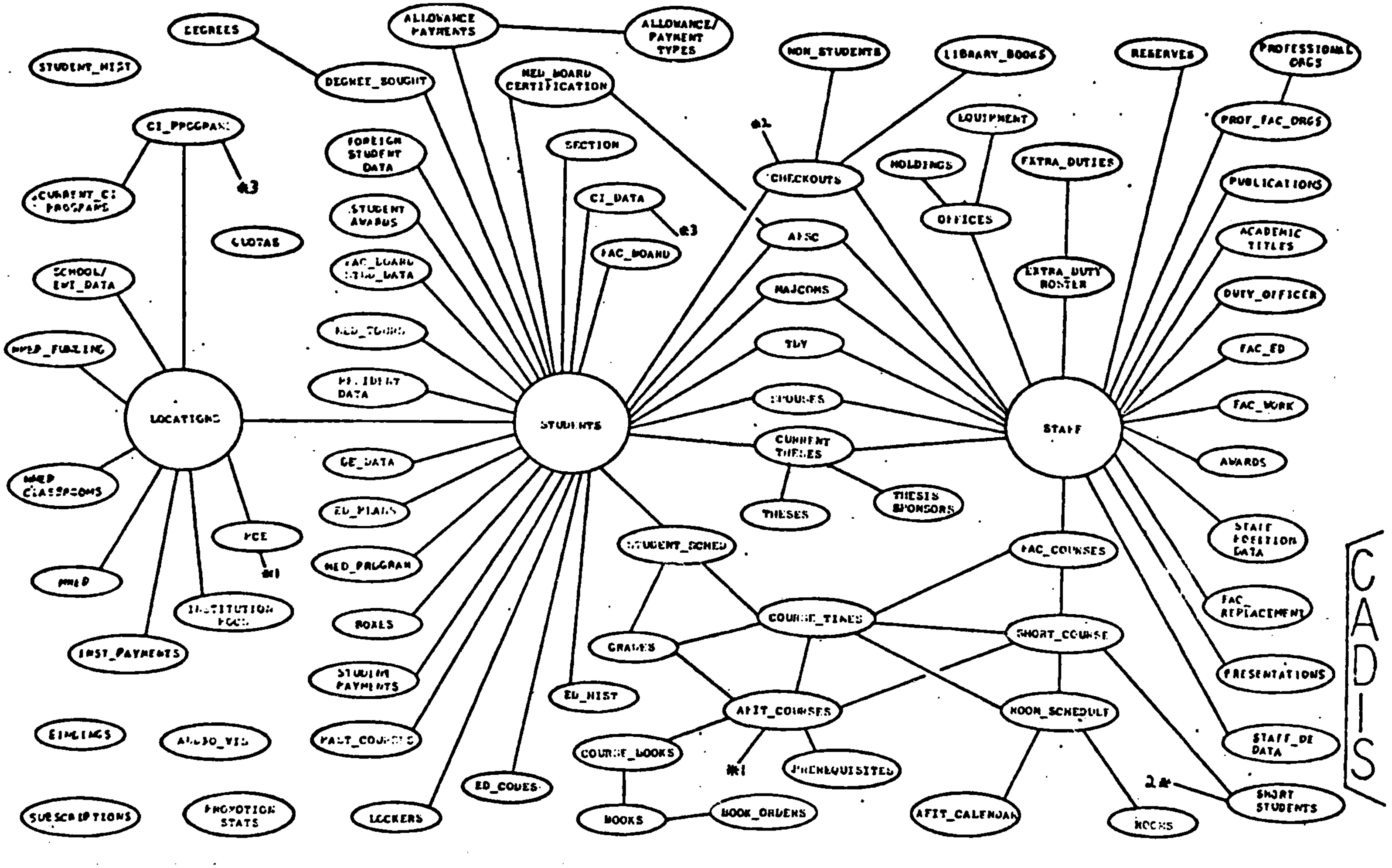
ADMISSIONS DATA: entry date, entry term, adm college, adm classification, adm degree, adm major, prev coll FICE code, prev college GPA, admission action (e.g. A=application process complete), admission type (e.g. GRD=entering with undergrad degree), prev academic level (e.g. BD=baccalaureate degree), entrance exam 1 (e.g. GMAT=GMAT total), test score 1, entrance exam 2, test score 2, entrance exam 3, test score 3, entrance exam 4, test score 4, entrance exam 5, test score 5, entrance exam 6, test score 6, entrance exam 7, test score 7, entrance exam 8, test score 8, skf maint date (date record last maintained)

STUDENT TERM FILE ATTRIBUTES AND ACADEMIC PROGRAM DATA: dean's list (e.g. *Y= on dean's list), academic action (e.g. *PW=placed on warning), withdraw code (e.g. A=academic difficulty), withdraw date, college, classification, primary major 1, primary major 2, primary major 3, primary minor 1, primary minor 2, secondary degree, secondary major, grad code (e.g. F=first degree program complete), mail building (e.g. EN=bldg 640), box number, reg type, prog notice flag (e.g. 2=student's program has been revised, a notice needs to be issued), grade rpt flag (e.g. 3=grade report received), term, social security no., last calc date (date of last system calculated grade), stf maint date, TRF earned hours (transfer credit awarded), curr earned hrs, curr qual hrs (does not include credit/no credit hrs), curr qual pts (current grade pts awarded for successful completion of academic work), cum earned hrs, cum qual hrs, cum qual pts, higher ed qual hours, higher ed qual points

REGISTRATION DATA (SPE-I): SPE crs number, SPE status (e.g. E=enrolled), grade type (e.g. CN=credit/no credit), official grade (e.g. A=excellent=4.0), prior grade (official grade before any grade change), registered hours, earned hrs flag, quality points, drop reason, quality hrs flag, career pointer (e.g. C=award graduate credit and distribute to graduate career), regristration level (e.g. G=graduate), session, SPE maint date (date of last change to SPE), course-college, course-col & dept, course-number, course-section

ADMINISTRATIVE TERM ACTIVITY DATA (SPE-II): SPE type, init career no (e.g. 2=graduate), init earned hours, init quality hours, init quality points, init higher ed qhrs, init higher ed qpts, SPE2 init maint date, TRF activity no, TRF school code (FICE code for transferring inst.), TRF hours earned, TRF quality hours, TRF quality points, TRF begin date (date transfer work began), TRF end date, TRF term applied (term to which transfer credit is to be applied), SPE2 TRF maint date, deg activity no, deg school code (FICE code for school awarding the degree), deg date, deg major 1, deg major 2, deg major 3, deg minor 1, deg minor 2, deg honors (e.g. M=magna cum laude) SPE2 deg maint date

Appendix IC


CADIS
STAFF
STUDENTS
LOCATIONS
DEGREES
STUDENT_HIST
ALLOWANCE PAYMENTS
ALLOWANCE/ PAYMENT TYPES
DEGREE_SOUGHT
MED_BOARD CERTIFICATION
NON_STUDENTS
LIBRARY_BOOKS
RESERVES
PROFESSIONAL ORGS
CI_PROGRAM
#3
#CURRENT_CI PROGRAMS
FOREIGN STUDENT DATA
SECTION
CI_DATA
#3
FAC_BOARD
#2
CHECKOUTS
EQUIPMENT
HOLDINGS
OFFICES
EXTRA_DUTIES
EXTRA_DUTY ROSTER
PROF_FAC_ORGS
PUBLICATIONS
ACADEMIC TITLES
DUTY_OFFICER
FAC_ED
FAC_WORK
AWARDS
STUDENT AWARDS
QUOTAS
FAC_BOARD STND_DATA
SCHOOL/ ENT_DATA
MED_TOURS
AFSC
MAJCOMS
TDY
SPOUSES
CURRENT THESES
THESES
THESIS SPONSORS
RESIDENT DATA
HPED_FUNDING
GE_DATA
ED_PLANS
MED_PROGRAM
BOXES
STUDENT PAYMENTS
PAST_COURSES
LOCKERS
ED_CODES
ED_HIST
STUDENT_SCHED
GRADES
COURSE_TIMES
AFIT_COURSES
COURSE_BOOKS
BOOKS
BOOK_ORDERS
PREREQUISITES
#1
FAC_COURSES
SHORT_COURSE
ROOM_SCHEDULE
AFIT_CALENDAR
ROOMS
2#
SHORT STUDENTS
STAFF_DE DATA
PRESENTATIONS
FAC_ REPLACEMENT
STAFF POSITION DATA
HPED CLASSROOMS
PCE
#1
HHED
INSTITUTION POCS
INST_PAYMENTS
EIMLIMGS
AUDIO_VIS
SUBSCRIPTIONS
PROMOTION STATS

Description of the Relations for the Consolidated AFIT Database

1. ACADEMIC_TITLES

   This relation contains a list of the Academic titles a faculty member may hold. Each title is indexed by a unique code which will allow other relations to key to the full title. Each entry has a maximum size of 35 characters.

2. AFIT_CALENDAR

   This relation allows users to maintain a list of the AFIT activities which occur throughout the year. Along with a description of each event is a special indicator to designate whether or not it is an activity which involves the commandant. Appropriate remarks may be included along with any POC (Point of Contact). Each entry has a maximum length of 128 characters.

3. AFIT_COURSES

   This relation contains information pertaining to all courses listed in the AFIT catalog. The course number (i.e. EE646A) will allow users to key into this information from other relations. Each entry has a maximum length of 145 characters.

4. AFSC

   This relation contains a list of the six character Air Force Specialty Codes (AFSC) and their descriptions. This medical code is strictly used by AFIT/CI. Each entry has a maximum length of 56 characters.

5. ALLOWANCE_PAYMENTS

   This relation contains data on the different allowances which have been paid to a non CI student. Each time a payment is made, the students SSAN will be entered along with the type of payment (books, thesis, etc) and the date on which the transaction occurred. Allowance type is a two letter code whose description is found in the ALLOWANCE/PAYMENT_TYPES relation. Each entry has a maximum length of 25 characters.

6. ALLOWANCE/PAYMENT_TYPES

   This relation contains the two letter Allowance_Types and their descriptions. Each entry has a maximum length of 52 characters.

7. AUDIO_VIS

This relation contains data on the audio/visual resources which may be checked out, such as movies and tapes, for which the AFIT library is responsible. Each entry has a maximum length of 139 characters.

8. AWARDS

This relation contains data concerning the various types of awards which have been presented to the staff. They include official awards and decorations, civilian and academic awards as well as resident school awards such as outstanding instructor. Each entry has a maximum length of 45 characters.

9. BINDINGS

This relation contains the special data required by the library for tracking periodicals that are bound together. Each entry has a maximum length of 46 characters.

10. BOOKS

This relation contains data for the bookstore and faculty and concerns the current availability of textbooks. Each entry has a maximum length of 124 characters.

11. BOOK_ORDERS

This relation contains information on the status of the orders placed by the bookstore for textbooks. Each entry has a maximum length of 105 characters.

12. BOXES

This relation maintains data on the availability and disposition of student mailboxes. When a box has been assigned, the students SSAN will be placed with the box number; when unassigned, the SSAN for a box will be empty (blank). This procedure will allow administrative personnel to not only assign multiple students to a box, but obtain a list of boxes with no students. Each entry has a maximum length of 13 characters.

13. CHECKOUTS

This relation contains the data essential for checking out material from the library. Id_No is the SSAN, or other unique identifier in the case of foreign students, of the student or staff member checking out the book. This will allow indexing into the STUDENT, STAFF, NON_STUDENT, SHORT_STUDENTS and LIBRARY_BOOKS relations. Each entry has a maximum length of 65 characters.

14. CI_DATA

This relation contains data peculiar to those students enrolled in civilian institutions. Because of the requirements dictated by the AFIT/CI mission, certain additional data must be maintained beyond the basic data found in the STUDENTS relation. The data in STUDENTS and CI_DATA are logically linked by the students SSAN. Each entry has a maximum length of 188 characters.

15. CI_PROGRAMS

This relation contains all AFIT/CI programs and their descriptions which are in the AFIT/CI course catalog. Each program is represented by a unique code which is the same as the Ed_Code used by the resident school. Each entry has a maximum length of 55 characters.

16. COURSE_BOOKS

This relation contains data on the required textbooks for currently scheduled courses. Data includes the number of books needed for the course. Each entry has a maximum length of 72 characters.

17. COURSE_TIMES

This relation contains the basic course scheduling data for all the resident schools. The Status and Wait_List_No would be used primarily by the registrar and represent the offering status (i.e. "open for registration") and the number of students on the waiting list for the course. Each entry has a maximum length of 37 characters.

18. CURRENT_CI_PROGRAMS

This relation contains data pertinant to those AFIT/CI programs which are currently active and have students enrolled in them. Each program is indexed to the LOCATIONS relation for institution name and address and contains a POC (Point of Contact) for the particular program. Each entry has a maximum length of 50 characters.

19. CURRENT_THESES

This relation contains data relevant only to those theses currently underway at AFIT, e.g. thesis number and thesis committee. Other information about the thesis itself is entered into the THESES relation and remains as a permanant record of the project. These two relations are connected by the Thesis_No attribute. Each entry has a maximum length of 51 characters.

20. DEGREES
This relation is a list of the various kinds of degrees which students and faculty may be (or have been) awarded. Each degree is identified by a unique five letter code which shows the level of the degree and the area of study (i.e. Masters Degree in Electrical Engineering = MSEE). Also associated with each code is a full title or description of the degree. Each entry has a maximum length of 55 characters.

21. DEGREE_SOUGHT
This relation provides data on the degree, major and minor, a current AFIT student is seeking. This includes both resident and CI students. Each entry has a maximum length of 64 characters.

22. DE_DATA
This relation contains student data peculiar to students attending the School of Civil Engineering (AFIT/DE). Because of the requirements dictated by AFIT/DE needs, certain additional student data must be maintainedd beyond the basic data found in the STUDENTS relation. The data in STUDENTS and DE_DATA are linked by the students SSAN. Each entry has a maximum length of 31 characters.

23. DUTY_OFFICER
This relation contains data on those permanant party officers who are eligible to be AFIT duty officers. The credit attribute is used to calculate when they will stand the duty, the Date attribute tells when they are scheduled for it and the Duty attribute indicates which list duty the officer is assigned, i.e. Captain or Major/Ltc. Data in the DUTY_OFFICER and STAFF relations are tied by the SSAN of the officer. Each entry has a maximum length of 25 characters.

24. ED_CODES
This relation contains data on valid educational codes and their descriptions. Each entry has a maximum length of 34 characters.

25. ED_HIST

This relation contains a completet list of the institutions a student has attended prior to arriving at his current AFIT tour. A description of the degree earned (if any) can be found in the DEGREES relation. A blank Degree_Code entry indicates no degree was achieved for the particular period denoted by Begin_Date through End_Date. Also included are any pertinant remarks. Each entry has a maximum length of 116 characters.

26. ED_PLANS

This relation contains the Education Plan of each resident student. This allows the tracking of the projected courses on a per quarter basis as well as the credit hours. It is assumed that only plans already approved by the department head will be entered. Each entry has a maximum length of 20 characters.

27. EQUIPMENT

This relation contains the necessary data to plan and track any orders for equipment, furniture or supplies in which the various departments within AFIT might be interested. Another possible use might be as an equipment inventory list. Each entry has a maximum length of 106 characters.

28. EXTRA_DUTIES

This relation is a list of all the extra duties to which a AFIT staff member might be assigned. Each duty title or description is identified by a unique six character code which also serves as an abbreviated form of the title. Each entry has a maximum length of 56 characters.

29. EXTRA_DUTY_ROSTER

This relation contains data on all personnel assigned to extra duties, the duty(s) to which they are assigned and the effictive dates. The Extra_Duty_Code keys into the EXTRA_DUTIES relation if the full length extra duty title is needed. Each entry has a maximum length of 21 characters.

30. FAC_BOARD

This relation contains general information pertaining to the reasons for and actions taken on a particular Faculty Board. The students SSAN will link this relation with additional information on the student. Sufficient fields are provided to allow previous board actions to be noted. Each entry has a maximum length of 207 characters.

31. FAC_BOARD_STUD_DATA
This relation contains the students academic information required by the members of a particular faculty board. A u[illegible]cord is created each time a board is held for a student, thus providing a journal of his progress. Each entry has a maximum length of 39 characters.

32. FAC_COURSES
This relation contains data on an instructors schedule for a given quarter. For those courses which have more than one instructor participating, the Instructor_No attribute identifies which instructor he is; e.g. "2" means second instructor or instructor number 2. Each entry has a maximum length of 28 characters.

33. FAC_ED
This relation represents an educational history of each faculty member. All attributes except SSAN are free-form text fields and could be indexed into the DEGREES relation if formatted appropriately. Special interests of instructors could also be listed, e.g. microcomputers. Each entry has a maximum length of 99 characters.

34. FAC_REPLACEMENT
This relation contains data on projected faculty replacements. Each replacement is uniquely identified by the appropriate position number (Pos_No). Each entry has a maximum length of 157 characters.

35. FAC_WORK
This relation represents the work history of each faculty member. The value of the attribute Rank will vary according to the particular type of employment and the given rank held at a particular time. Thus, if two different ranks were held at the same place but at different times, both work experiences would be uniquely identified and entered. Each entry has a maximum length of 55 characters.

36. FOREIGN_STUDENT_DATA
This relation contains data unique to those students who are members of foreign Allied Armed Forces such as country and funding source. The basic student data is

kept in the STUDENTS relation. Proper use of this relation assumes that a method of identifying foreign students, similar to the SSAN, will be implemented within AFIT. Each entry has a maximum length of 41 characters.

37. GRADES
This relation contains data on the grades for each course in which a resident student is enrolled. Grades could be kept on a quater by quarter basis only or retained for the duration of a students AFIT assignment. Each entry has a maximum length of 21 characters.

38. HOLDINGS
This relation is used by the library to track which office has what library material checked out. Each entry has a maximum length of 46 characters.

39. INSTITUTION_POCS
This relation contains the POCs (Points of Contact) for each civilian institution. Generally this will be the registrar or admissions office as opposed to the POCs for a particular program as in the CURRENT_CI_PROGRAMS relation. Further information concerning the institution may be found in the LOCATIONS relation. Each entry has a maximum length of 45 characters.

40. INST_PAYMENTS
This relation will enable AFIT/CI to maintain a record of the types and dates of payments made to civilian institutions. The type of payment is represented by a four character code which in indexed into the ALLOWANCE/PAYMENT_TYPES relation. Further information about the institution can be found in the LOCATIONS relation. Each entry has a maximum length of 23 characters.

41. LOCATIONS
This relation contains the code, name and address, on all locations of interest to the users of the database. LOCATIONS will include military installations, government facilities, civilian institutions and Education With Industry locations. Any relation requiring additional information about a location may index into this relation by using the attribute Location_Code. Each entry has a maximum length of 85 characters.

42. LOCKERS

This relation maintains data on the availability and disposition of student lockers. When a locker has been assigned, the students SSAN will be paired with the locker number; when unassigned the SSAN attribute for a locker tuple will be empty (blank). This will allow all unassigned lockers to be identified. Each entry has a maximum length of 13 characters.

43. LIBRARY_BOOKS

This relation contains a list of all the publications in the library which are available for checkout by the students and faculty. Each entry has a maximum length of 80 characters.

44. MAJCOMS

This relation contains a list of the major air command abbreviations and their full title (i.e. SAC = Strategic Air Command). Each entry has a maximum length of 35 characters.

45. MED_BOARD_CERTIFICATION

This relation is comprised of a students medical specialty code, his individual scores for medical certification tests, the dates of the certification and the actual test scores. Since the medical specialty code is an AFSC, a description of the specialty can be found in the AFSC relation. Each entry has a maximum length of 48 characters.

46. MED_PROGRAM

This relation contains data peculiar to those students who are enrolled in a medical program. It can be tied to the student relation via SSAN. Each entry has a maximum length of 48 characters.

47. MED TOURS

This relation contains a list of the assignments a medical student has accumulated. Each entry has a maximum length of 32 characters.

48. MMEP

The data in this relation pertains to the AF bases & supporting institutions which constitute the MinuteMan Education Program. Further information about the supporting institution and base can be found in the LOCATIONS relation. Each entry has a maximum length of 10 characters.

49. MMEP_CLASSROOMS
This relation is a list of the rooms and buildings used at an installation for the MMEP program. Each entry has a maximum length of 12 characters.

50. MMEP_FUNDING
This relation contains a list of the AF MMEP base locations and the funding source of the program along with the number of students enrolled. Each entry has a maximum length of 10 characters.

51. NON_STUDENTS
The purpose of this relation is to track those people who are not enrolled in AFIT but who, for some reason, are authorized to use the AFIT library. The attribute Card_No will contain the users SSAN if library cards are not used. Each entry has a maximum length of 69 characters.

52. OFFICES
This relation consists of office symbols and their corresponding department names. Each entry has a maximum length of 34 characters.

53. PAST COURSES
The non-AFIT courses which a student has previously taken but which did not lead to a completed degree could be recorded in this relation. Each entry has a maximum length of 41 characters.

54. PCE
This relation contains data concerning non-resident Professional Continuing Education (PCE) courses which are offered at various locations under the auspices of AFIT. Additional data such as the maximum number of students allowed in the course and the actual number enrolled can also be maintained. Further information about a course can be obtained by indexing into the AFIT_Courses relation. Each entry has a maximum length of 21 characters.

55. PREREQUISITES
The prerequisite courses for any given course are contained in this relation. Because only a single course

and one prerequisite are listed together, the user is cautioned to perform a thorough search of the relation by properly constructing his query to insure that all required data is retrieved. Each entry has a maximum length of 12 characters.

56. PRESENTATIONS
Data on the presentations which a faculty member gives is contained in this relation. The attribute Location is a text field and is not tied to the LOCATIONS relation since many unique and unusual presentation location are possible. Additionally, the attribute Organization denotes the organization to which the presentation was given. Each entry has a maximum length of 140 characters.

57. PROFESSIONAL_ORGS
This relation contains general data on professional organizations, and their full names, to which faculty and staff members belong. A unique identifier code is used to distinguish between the organizations. Each entry has a maximum length of 55 characters.

58. PROF_FAC_ORGS
This relation contains specific data pairing a given faculty of staff member with a particular professional organization code, indicating that he is a member of the associated professional organization. The attribute Org_Abbrev is used to obtain additional information about the organization by keying into the PROFESSIONAL_ORGS relation. Each entry has a maximum length of 14 characters.

59. PROMOTION_STATS
This relation is to be used by AFIT/PA to analyse and track the results of the various AF boards in which AFIT graduates are considered for promotion. The attribute Board_Kind will be determined by the purpose of the particular board (i.e. Capt, Maj, etc). Each entry has a maximum length of 26 characters.

60. PUBLICATIONS
This relation contains data on the articles and books which the AFIT faculty publishes. Each entry has a maximum length of 190 characters.

61. QUOTAS

This relation will be used by AFIT/ED to keep track of the estimated and actual quotas of students in each educational code. Each entry has a maximum length of 20 characters.

62. RESERVES

This relation contains data on books and publications which are placed on reserve by members of the faculty. Each entry has a maximum length of 35 characters.

63. RESIDENT DATA

This relation contains student data peculiar to AFIT resident students. Data in this relation was only identified as necessary by AFIT/EN and AFIT/LS. Information in the STUDENTS and RESIDENT_DATA relations are logically connected by the students SSAN. Each entry has a maximum length of 98 characters.

64. ROOMS

This relation is a list of the rooms within AFIT which can potentially be scheduled for various activities. The attribute Special_Room_Type is used to denote such things as a secure room, auditorium, laboratory or any other room with special features. Each entry has a maximum length of 11 characters.

65. ROOM SCHEDULE

This relation allows the centrally schedulable rooms at AFIT to be managed in order to forcast and track their use and thus eliminate conflicts. Each entry has a maximum length of 46 characters.

66. SCHOOL/EWI_DATA

This relation contains data which pertains to civilian institutions and Education With Industry sites for which AFIT/CI is responsible. Each entry has a maximum length of 17 characters.

67. SECTION

This relation contains data on current AFIT resident school sections. The STUDENTS relation can be logically related to this relation by either the attribute Section or Leader_SSAN. Each entry has a maximum length of 41 characters.

68. SHORT_COURSE
This relation contains data on all short courses taught at AFIT. Each course is uniquely identified by the six character attribute Course. Course can also be used to index into the AFIT_Courses relation. Each entry has a maximum length of 109 characters.

69. SHORT_STUDENTS
This relation functions similarly to the relation STUDENTS but only maintains data on non-resident students such as part-time or short-course students. The attribute Course' will allow this relation to tie into the AFIT_COURSES relation for more information on the course itself. Each entry has a maximum length of 90 characters.

70. SPOUSES
This relation allows information about the spouse of both students and faculty to be retained. The attribute SSAN allows this relation to be logically related to both the STAFF and STUDENTS relation. The attribute Spouse_Type distinguishes between staff and students spouses and the attribute Military_Spouse signifies a spouse who is also in the military. Each entry has a maximum length of 49 characters.

71. STAFF
This relation is the primary repository for all personal data on the faculty and staff of AFIT. Further information on most of the attributes can be obtained from other relations as required by indexing into them with the appropriate common attribute. Each entry has a maximum length of 232 characters.

72. STAFF_DE_DATA
This relation contains staff data peculiar to AFIT/DE. It will tie into the STAFF relation by means of the SSAN of the staff member. Each entry has a maximum length of 15 characters.

73. STAFF_POSITION_DATA
This relation contains data on the education levels associated with staff positions. Tuples are keyed by position number and data is given on ed codes and degree levels, both required and assigned. A tie into the STAFF relation is possible via SSAN. Each entry has a maximum length of 26 characters.

74. STUDENTS

This relation is where the common data concerning all students other than part-time and short-course students is found. These attributes are those most commonly requested by all the AFIT departments. When the attribute Foreign_Student_Ind is an "F", a foreign student is denoted and additional data about him is in the FOREIGN_STUDENTS relation. The logical link between these two relations is the students SSAN. For all students the attribute Duty_Location will contain the location code for The school to which they are assigned. EN and LS would have seperate location codes. Additionally, the attributes GRE and GMAT only contain the total scores for that test; a further breakdown may be obtained from the registrar. Each entry has a maximum length of 229 characters.

75. STUDENT_AWARDS

This relation concerns the various types of awards which have been accorded to the AFIT students. They may include official decorations and scholastic awards such as Dean's List. Each entry has a maximum length of 45 characters.

76. STUDENT_HIST

This relation could contain the data which is desired to be kept on previous AFIT students. The attribute SSAN serves as the only key although a secondary index on Name might be possible. Name is not intended as a logical link into any other relation. However, additional data on some of the other attributes may be obtained as required by using other logical links. Each entry has a maximum length of 143 characters.

77. STUDENT_PAYMENTS

This relation contains a record of the payments made by AFIT/CI to the students enrolled in their programs. The attribute Payment_Type is the same as the attribute Allowance_Type in the ALLOWANCE_PAYMENTS relation. Further information about this attribute is contained in the ALLOWANCE/PAYMENT_TYPES relation. Each entry has a maximum length of 40 characters.

78. STUDENT_SCHED

This relation is updated each quarter from the registrars office and contains the current list of courses in which a student has oficially enrolled. By logically linking with the COURSE_TIMES relation, a students schedule can be determined. The logical link to

Room_Schedule could also be followed to find a student during an emergency while he was in class. Each entry has a maximum length of 15 characters.

79. SUBSCRIPTIONS
This relation contains data on the periodicals to which the library subscribes. Each entry has a maximum length of 147 characters.

80. TDY
This relation contains information concerning the TDY trips taken by both staff and students. The attributes will not only allow for recording projected trip costs but also the actual cost once the trip has been completed. Each entry has a maximum length of 81 characters.

81. THESES
This relation contains the general information required on all AFIT theses. It will provide a library of data about completed theses thus allowing quick and easy access to historical data. Current theses will be listed here also and logically linked to the CURRENT_THESES relation by the attribute Thesis_No. The CURRENT_THESES relation is concerned only with on going projects. Once a current thesis is completed it is removed from CURRENT_THESES and the only record of it will remain in this relation. Each entry has a maximum length of 159 characters.

82. THESIS_SPONSORS
This relation contains a list of sponsors of a resident AFIT thesis along with their location. Further information about the location can be found in the LOCATIONS relation. Each entry has a maximum length of 60 characters.

The following relations pertain strictly to the registrars function at AFIT. After examing the contents, structure and intent of the current data system in use at AFIT/RR (INTERACT), these relations were designed to work in concert with the above relations yet still allow the functions currently being performed to be fulfilled. It is felt that with these basic relations and the capabilities of a good

DBMS the registrars requirements and functions can be included with only a relatively few applications programs and no loss of capability.

Because present situations may preclude the registrar from being included into a single consolidated AFIT database at this time, the above relations were designed with their exclusion being a definite possibility. Therefore, if and when the registrar is included, some of the previous relations could be reduced in size to eliminate the small amount of overlap which would occur.

83. COURSE_GRADES
    This relation would be the permanent grade record of each course taken by a student. All of these records for a student would comprise a major portion of his transcript. Each entry has a maximum length of 58 characters.

84. COURSE_STATS
    This relation contains statistics for each course currently offered within AFIT and it is assumed they are cumulative. However, if this is not the desired method, the addition of the attributes Quarter and Year as keys will eliminate any duplicate entries and allow data to be maintained for each quarter. Each entry has a maximum length of 24 characters.

85. GMAT_SCORES
    This relation contains the total GMAT score for a particular student. A GMAT indicator and the students SSAN could serve as a logical link into this relation. Each entry has a maximum length of 12 characters.

86. GRE_SCORES
    This relation contains a breakdown of the GRE scores for a particular student. If this relation is incorporated the total score will no longer have to be kept seperate in other student relations, it can be computed as required. A GRE indicator and the students SSAN could be used to logically link into this relation. Each entry has a maximum length of 18 characters.

87. RR_DATA
    This relation contains student data specifically required by the registrar. The attributes GRE_Ind and GMAT_Ind are used to denote which set of tests the

student took; the scores can be found by indexing into the appropriate relation using the students SSAN. Each entry has a maximum length of 77 characters.

88. TRANSFERS
This relation contains the data required for transfer students. Each entry has a maximum length of 48 characters.

RELATIONS FOR THE CONSOLIDATED AFIT DATABASE
KEYS FOR EACH RELATION ARE DENOTED BY "*"s

1. ACADEMIC_TITLES:
Academic_Title_Code*, Academic_Title.

2. AFIT_CALENDAR:
Day*, Month*, Year*, Starting_Time*, Activity, CC_Activity, Room*, Building*, POC, Remarks.

3. AFIT_COURSES:
Course*, Title, Description, Min_Credit, Max_Credit, Special_Grading, Section_Control, Remarks, Lec_Hrs, Lab_Hrs, Student_Limit, Special_Room_Type.

4. AFSC:
AFSC*, Description.

5. ALLOWANCE_PAYMENTS:
SSAN*, Allowance_Type*, Date_Paid*, Amount.

6. ALLOWANCE/PAYMENT_TYPES:
Allowance_Type*, Description.

7. AUDIO_VIS:
Title*, Time, Producer*, Speaker, Subject, Production_Date.

8. AWARDS:
SSAN*, Award, Date*.

9. BINDINGS:
Title*, Volume, Month, Year, Color_No, Letter_Color.

10. BOOKS:
Title*, Author*, Publisher, No_Avail, Price, Office.

11. BOOK_ORDERS:
Author, Title, Order_No*, No_Ordered, Date, Status.

12. BOXES:
SSAN*, Box_No*.

13. CHECKOUTS:
Call_No*, Patron_Name, ID_No*, Due_Date.

14. CI_DATA:
SSAN*, Program, Output_AFSC, Degree_Code, Prior_GPA, Available_Date, Prior_AFIT, Departure_Reason, Suspense_Date, Legal_Residence, Thesis_Pay, Suspense_Action, Status, Begin_Date, C_GPA, Term_GPA.

15. CI_PROGRAMS:
Program*, Description.

16. COURSE_BOOKS:
Course*, Author*, Title*, No_Required, Quarter*, Year*.

17. COURSE_TIMES:
Course*, Starting_Time*, Day*, Quarter*, Year*, Room, Building, Course_Status, Wait_List_No, Begin_Date, End_Date.

18. CURRENT_CI_PROGRAMS:
Location_Code*, Program*, POC, DPhone.

19. CURRENT_THESES:
SSAN*, Thesis_No, Advisors_SSAN, Reader1, Reader2.

20. DEGREES:
Degree_Code*, Description.

21. DEGREE_SOUGHT:
SSAN*, Degree_Code, Major, Minor.

22. DE_DATA:
SSAN*, DAFSC, Funct_Acct_Code, DOS, Arrival_Date.

23. DUTY_OFFICER:
SSAN*, Duty, Credit_Date, Duty_Date.

24. ED_CODES:
Ed_Code*, Ed_Code_Title.

25. ED_HIST:
SSAN*, College*, Begin_Date, End_Date, Degree_Code, Departure_Reason, Remarks.

26. ED_PLANS:
SSAN*, Course*, Quarter*, Year*, Credits.

27. EQUIPMENT:
Stock_No*, Description, Price, Date_Ordered*, Date_Received, Availability, Office.

28. EXTRA_DUTIES:
Extra_Duty_Code*, Extra_Duty_Title.

29. EXTRA_DUTY_ROSTER:
SSAN*, Extra_Duty_Code*, Date.

30. FAC_BOARD:
SSAN*, Trouble_Term, Quarter*, Year*, Trouble_Year, Reason, Initial_Analysis, Board_Action, Success, Remarks.

31. FAC_BOARD_STUD_DATA:
SSAN*, Quarter*, Year*, UG_CGPA, GRE, GMAT, Term_GPA, C_GPA, A_Cred, B_Cred, C_Cred.

32. FAC_COURSES:
SSAN*, Course*, Quarter*, Year*, Instructor_No, Group_Contact_Hrs, Ind_Contact_Hrs.

33. FAC_ED:
SSAN*, Primary_Ed_Level, Secondary_Ed_Level, Special_Interest.

34. FAC_REPLACEMENT:
Pos_No*, Office, DAFSC, Rank, Name, Arrival_Date, Degree_Required, Academic_Specialty, Status, DPhone.

35. FAC_WORK:
SSAN*, Location*, Rank*, Begin_Date, End_Date.

36. FOREIGN_STUDENTS_DATA:
ID_No*, Service, Country, Funding.

37. GRADES:
SSAN*, Course*, Quarter*, Year*, Grade.

38. HOLDINGS:
Title*, Office*, Begin_Date, End_Date.

39. INSTITUTION_POCS:
Location_Code*, POC, DPhone.

40. INST_PAYMENTS:
Location Code*, Date*, Payment Type*, Amount.

41. LOCATIONS:
Location Code*, Location Name, Local Address.

42. LOCKERS:
SSAN*, Locker No*.

43. LIBRARY_BOOKS:
Call_No*, Title, Author.

44. MAJCOMS:
MAJCOM*, MAJCOM_Title.

45. MED_BOARD_CERTIFICATION:
SSAN*, Med_Spec_Code*, Certification_Date, ANAT_Score, PHYS_Score, BIOCH_Score, PATH_Score, MICRO_Score, PHARM_Score, BEHSCI_Score, Score_Date.

46. MED_PROGRAM:
SSAN*, MCAT_Score, MCAT_Score_Date, Degree_Code, Med_Program, Accession_Source.

47. MED_TOURS:
SSAN*, Med_Spec_Code, Location_Code*, Begin_Date*, End_Date.

48. MMEP:
Location_Code*, Supporting_Inst_Code.

49. MMEP_CLASSROOMS:
Location_Code*, Room*, Building*.

50. MMEP_FUNDING:
Location_Code*, Fund_Type*, No_Students.

51. NON_STUDENTS:
Name, Card_No*, Organization, DPhone.

52. OFFICES:
Office*, Office_Title.

53. PAST_COURSES:
SSAN*, College*, Course*, Grade, Quarter*, Year*.

54. PCE:
Course*, Location_Code*, MAJCOM*, Student_Limit, No_Students.

55. PREREQUISITES:
Course*, Prerequisite*.

56. PRESENTATIONS:
SSAN*, Date*, Title, Location, Organization*, Subject.

57. PROFESSIONAL_ORGS:
Org_Abbrev*, Org_Name.

58. PROF_FAC_ORGS:
    SSAN*, Org_Abbrev*.

59. PROMOTION_STATS:
    Board_Date*, Board_Kind*, USAF, AFIT, USAF_Elig, AFIT_Elig.

60. PUBLICATIONS:
    SSAN*, Title*, Date, Subject, Source*, Location, Pub_Kind, Co_Auth_Name.

61. QUOTAS:
    Ed_Code*, Quota_Type*, YR1, YR2, YR3, YR4, YR5.

62. RESERVES:
    Call_No*, FAC_SSAN*, Date_Reserved*.

63. RESIDENT_DATA:
    SSAN*, Last_Organization, Duration, DOC, Emergency_Address.

64. ROOMS:
    Room*, Building*, Max_Size, Special_Room_Type.

65. ROOM_SCHEDULE:
    Day*, Month*, Year*, Starting_Time*, Ending_Time, Room, Building*, Function.

66. SCHOOL/EWI_DATA:
    Location_Code*, Servicing_AFO, Tuition_Rate.

67. SECTION:
    Section*, Graduation_Date, Entry_Date, No_Students, Leader_SSAN, Advisors_SSAN, No_Non_AF.

68. SHORT_COURSE:
    Course*, Title, Begin_Date*, End_Date, Room*, Building*, Starting_Time, Description.

69. SHORT_STUDENTS:
    SSAN*, Course*, Begin_Date*, Name, Rank, Organization, DPhone

70. SPOUSES:
    Sponsors_SSAN*, Name, DOB, No_Deps, Spouse_Type, Military_Spouse.

71. STAFF:
    SSAN*, Name, Rank, HPhone, DPhone, Room, Office, Pos_No, Duty_Title, Date_Assigned, Departure_Date, Local_Address, DOB, DOR, Acad_Title_Code, DOAR, Marital_Status, Sex, PAFSC, DAFSC, Ethnic_Group,

Service, Photo_Date, Security_Clearance, TMST, Previous_MAJCOM.

72. STAFF_DE_DATA:
SSAN*, Course_Directed*.

73. STAFF_POSITION_DATA:
Pos_No*, SSAN, Ed_Code_Required, Ed_Code_Assigned, Degree_Level_Required, Degree_Level_Assigned.

74. STUDENTS:
SSAN*, Name, Rank, PAFSC, Ed_Code, DOR, Sex, DOB, Advisors_SSAN, Entry_Date, DPhone, HPhone, Local_Address, Foreign_Service_Ind, Service, Departure_Date, Security_Clearance, Duty_Location, Aero_Rating, Previous_MAJCOM, Marital_Status, TMST, Ethnic_Group, GRE, GMAT, Duty_Title, Section, Academic_Standing.

75. STUDENT_AWARDS:
SSAN*, Award, Date*.

76. STUDENT_HIST:
SSAN*, Name, Rank, PAFSC, Ed_Code, Sex, Entry_Date, Departure_Date, Service, Local_Address, Address_Date, Aero_Rating, Marital_Status, Ethnic_Group, GRE, GMAT, Degree_Code.

77. STUDENT_PAYMENTS:
SSAN*, ESA_No*, Date*, Payment_Type*, Amount.

78. STUDENT_SCHED:
SSAN*, Course*.

79. SUBSCRIPTIONS:
Title*, Check_In, Renewal_Date, Price, Expiration_Date, Requester, Begin_Date, Follow_Up_Date, Follow_Up_Remarks.

80. TDY:
Destination*, SSAN*, Departure_Date*, Return_Date, Estimated_Cost, Actual_Cost,Fees, Per_Diem, Travel.

81. THESES:
Thesis_No*, Title, Thesis_Sponsor, Classification, First_Author, Second_Author, Subject, Thesis_Pub_Date, DTIC_No, Cleared_For_Release.

82. THESIS_SPONSORS:
Thesis_Sponsor*, Location.

RELATIONS PERTAINING TO THE REGISTRARS INFORMATION

83. COURSE_GRADES:
SSAN*, Course*, Grade_Type, Grade, Prior_Grade, Registered_Hrs, Earned_Hrs, Qual_Pts, Drop_Reason, Date.

84. COURSE_STATS:
Course*, Proj_Enrollment, Min_Enrollment, No_Enrolled, Demand, Drops, Wait_List_No, No_Not_Registered, No_Added, No_Audits.

85. GMAT_SCORES:
SSAN*, Total.

86. GRE_SCORES:
SSAN*, Verbal, Quantitative, Total.

87. RR_DATA:
SSAN*, Manning_Code, Output_MAJCOM, UG_CGPA, Degree_Code, Prior_AFIT, Folder_Location, Departure_Reason, Admission_Action, Transfer_Ind, Prev GPA, C_GPA, GRE_Ind, GMAT_Ind, Re Admit Term, Academic Elig, Calc Required Term, Degree_Checkout, Admin_Hold, Re_Admit_Year, Calc Required_Year.

88. TRANSFERS:
SSAN*, Location_Code*, TRF_Hrs, TRF_Qual_Hrs, TRF Qual Pts, Begin Date, Quarter, TRF In Date, TRF End Date, Old_Code.

Appendix I,F

ATTRIBUTE	RELATIONS	DESCRIPTION	EXAMPLE	SIZE

1. Academic_Elig	87	Indicates whether the student is eligible to register for the current program in subsequent terms	Y (yes, automatic);
			N (no, automatic);
			YM (yes manual);
			NM (no manual)
				2

2. Academic_Specialty	34	Text field indicating faculty members academic specialty	ARTIFICIAL INTELLIGENCE	30
3. Academic_Standing	74	Indicates whether the student is on probation or not	Y (yes); N (no)	1
4. Academic_Title	1	A faculty members actual academic title	PROFESSOR	30
5. Academic_Title_Code	1*, 71 (see note 1)	Short version of a full academic title, keys to the long version	PROF	5
6. Accession_Source	46	Source from which the student entered a medical program	USAFA; USAA; AFROTC; CIV; USNA; ADAF	6
7. A_Cred	31	Total number of credit hours a student has with an "A"	12	2
8. Activity	2	Free form text field listing an AFIT calendar event	COMMANDERS CALL	30
9. Actual_Cost	80	Actual cost of a completed TDY	697.58	7
10. Address_Date	76 (see note 2)	Date the address was last confirmed	291082	6
11. Admin_Hold	87	Indicates whether a students records are being held	" " (no); "H" (yes) (see note 3)	1
12. Admission_Action	87	Indicates whether a student has been formally admitted to AFIT	" " (not a part time student); "I" (incomplete application, admitted); "C" (complete application, admitted)	1
13. Advisors_SSAN	19, 67, 74	In relation 19 it is the thesis advisors SSAN in others it is the faculty advisors SSAN	521708233	9

14. Aero_Rating	74, 76	Indicates a persons aeronautical rating	"U" (unknown);
			"P" (pilot);
			"N" (navigator);
			"O" (other);
			" " (none)
				1

15. AFIT	59	Percentage of AF officers who have attended AFIT who were promoted by a particular board	97.5	4
16. AFIT_Elig	59	The number of AF officers who have attended AFIT and were eligible for a particular board	1132	4
17. AFSC	4*	AF Specialty Code; includes the 4 digit code plus the one letter prefix and suffix	T5135B	6
18. Allowance_Type	5*, 6*	A code which indicates the type of allowance paid	"BK" (book)	2
19. Amount	5, 40, 77	The amount of a payment	99999.99	8
20. ANAT_Score	45	Medical Board Anatomy test score	500	3
21. Arrival_Date	22, 34	In relation 22 it is the arrival date of an AFIT/DE student In relation 34 it is the arrival date of a faculty replacement	291082	6
22. Author	10*, 11, 16*, 43	Author of a particular work	ARTHUR C. CLARK	30
23. Availability	27	Free text field for comments on the availability of an item	SHIPPED	20
24. Available_Date	14	Projected availability date of an AFIT/CI student (out of program date)	281183	6

Appendix I,F

25. Award	8,75	Name of the award an individual has been given. Deans list could be included.	AF COMM MEDAL	30
26. B_Cred	31	Total number of credit hours a student has with a "B"	24	2
27. Begin_Date	14,17,25,35, 38,47*,68*, 69*,79,85	In relation 68 it distinguishes between different offerings of the same short course In relation 79 it is the subscription start date	50184	6
28. BEHSCI_Score	45	Med Board Behavioral Science Score	250	3
29. BIOCH_Score	45	Med Board Biochemistry Score	250	3
30. Board_Action	30	Free form text field for listing the requirements/ expectations resulting from a faculty board	CONTINUE AT AFIT	30
31. Board_Date	59*	Date of a particular promotion board	190683	6
32. Board_Kind	59*	Type of a particular promotion board	CAPT	4
33. Box_No	12*	Number of a student mail box	4132	4
34. Building	2*,17,49*,64*, 65*,68*	Building number that a particular room is in	640	3
35. Calc_Required_ Term	87	Indicates the term that a grade calculation is required for RR	W (winter)	2
36. Calc_Required_ Year	87	Indicates the year that a grade calculation is required for RR	82	2
37. Call_No	13*,43*,62*	Library card catalog call number	QA 76.9 .D3 H88	20
38. Card_No	51*	Unique library card number for a non-student	7000	4
39. CC_Activity	2	Indicates an activity in which the Commandant is involved	C (CC activity); " " (non-CC activity)	1
40. C_Cred	31	Total number of credit hours a student has with "C"	4	2
41. Certification_ Date	45	Date of a particular Med Board Certification	90153	6
42. C_GPA	14,31,87	Cumulative grade point average	3.35	4
43. Check_In	79	Date first copy of a given periodical arrived	140881	6
44. Classification	81	Security classification of a thesis	TSNF (top secret, noforn)	4
45. Cleared_For_ Release	81	Date a thesis was cleared for release	210779	6
46. Co_Auth_Name	60	Name of a co-author of a particular publication	HENRY POTOCZNY	30
47. College	25*,53*	Name of a college	UNIVERSITY OF KANSAS	20
48. Color_No	9	Color number of a library binding	8884 (rust)	4
49. Country	36	Name of a country	EGYPT	25
50. Course	3*,16*,17*, 26*,32*,37*, 53*,54*,55*, 68*,69*,78*, 83*,84*	For relation 53 it is a course number for any course at any college For all other relations it is the AFIT course number for an AFIT course only	EE799A	6
51. Course_Directed	72*	Alphanumeric designation for the course a DE faculty member directs	MGT001	6
52. Course_Status	17	Indicates whether a course is still open for students to register	O (open); C(closed); X (canceled)	1
53. Credit_Date	23	Date used to calculate when a person is next eligible to be duty officer	130581	6
54. Credits	26	Total number of credit hours a course is being taken for by a given student	4	1
55. DAFSC	22,34,71	Duty Air Force Specialty Code 1 char. prefix, 4 digits, 1 char. suffix	T5135B	6
56. Date	8*,11,29, 40*,56*,60,	A calendar date	210478	6

Appendix I,F

	75*,77*,83			
57. Date_Assigned	71	Date a person was assigned to a staff position	011280	6
58. Date_Ordered	27*	Date a piece of equipment was ordered	021065	6
59. Date_Paid	5*	Date an allowance was actually paid	031182	6
60. Date_Received	27	Date a piece of equipment was received	041282	6
61. Date_Reserved	62*	Date a book was put on an instructors reserve shelf	050860	6
62. Day	2*,17*,65*	Day of the week	M (monday)	1
63. Degree_Checkout	87	Term and year in which the student is expected to tomplete his degree requirements	W82	3
64. Degree_Code	14,20*,21, 25,46,76,87	Short version of a degree. Keys to long version	MSCS	5
65. Degree_Level_ Assigned	73	Degree level of staff member assigned against A particular staff position number	M (masters)	1
66. Degree_Level_ Required	73	Degfee level required for a particular staff position number	P (Phd)	1
67. Degree_Required	34	Long version of a degree	PHD, PULSE POWER	30
68. Demand	84	Total number of students desiring to register in a particular course	99	2
69. Denarture_Date	71,74,76,80*	Date a person departed AFIT	010683	6
70. Departure_Reason	14,25,87	Reason a person departed AFIT	GRADUATED	20
71. Description	3,4,6,15, 20,27,68	Free text field for describing an item	COMPUTER CARDS, PUNCH	50
72. Destination	80*	Name of a TDY destination	HQ USAF/ACD PENTAGON	25
73. DOAR	71	Date of Academic Rank, i.e., date of last academic promotion	081053	6
74. DOB	70,71,74	Date of Birth	081053	6
75. DOC	63	Date of Commissioning	053081	6
76. DOR	71,74	Date of Rank	053081	6
77. DOS	22	Date of separation, i.e., date a person will leave federal service	053099	6
78. DPhone	18,34,39,51, 69,71,74	Duty Phone Number	8048272887	10
79. Drop_Reasor	83	Free text field indicating why a student dropped a particular course	ADMINISTRATIVE ERROR	20
80. Drops	84	Total number of students who have dropped a given course	10	2
81. DTIC_No	81	Defense Technical Information Center Number	ADA082985	10
82. Due_Date	13	Date an item checked out of the library is due	111282	6
83. Duration	63	Length of a persons last tour in years	1.5	3
84. Duty	23	Indicates which duty list a particular duty officer is on	MLTC (major/LTC list); CAPT (captain)	4
85. Duty_Date	23	Indicates next date a particular person is duty officer	291182	6
86. Duty_Location	74	Location code signifying duty location of a student. (Separate codes for EN & LS)	12345	5
87. Duty_Title	71,74	General title a faculty member was hired under	PROFESSOR OF MATHEMATICS	30
88. Earned_Hrs	83	A number equal to the number of hours attempted (RR data)	16	2
89. Ed_Code	24*,61*,74, 76	4 letter Air Force Education Code	OCYY	4

90. Ed_Code_Assigned	73	Ed Code of the person assigned against a particular staff position number	OCTY	4
91. Ed_Code_Required	73	Ed Code required for a person assigned against a given staff position number	OCTY	4
92. Ed_Code_Title	24	Full english version of an Ed Code	COMPUTER TECHNOLOGY, GENERAL	30
93. Emergency_Address	63	Address of person to be contacted in the event of an emergency happening to a student	1023 NIDDY DR. WPAFB, OH. 45433	50
94. End_Date	17,25,35, 38,47,68	Date a course, etc., ended or will end	150683	6
95. Ending_Time	65	Time a room becomes available (24 hr clock)	1600	4
96. Entry_Date	67,74,76	Date a section or student started AFIT	010881	6
97. ESA_No	77*	Educational Service Agreement Number	F3360075A0569	13
98. Estimated_Cost	80	Estimated total cost of a TDY	9999.99	7
99. Ethnic_Group	71,74,76	Indicates a persons ethnic origin	W (white)	1
100. Expiration_Date	79	Date a library subscription expires	270985	6
101. Extra_Duty_Code	28*,29*	Unique short version of an extra duty title. Keys to unique longer version	CRPNAL	6
102. Extra_Duty_Title	28	Unique long english version of an extra duty title	COMPUTER RESOURCE PANEL MEMBER AT LARGE	50
103. FAC_SSAN	62*	Faculty members SSAN	444556666	9
104. Fees	80	Total amount of TDY money spent on fees	19.99	7
105. First_Author	81	SSAN of first author listed on a thesis	555667777	9
106. Folder_Location	87	Location of a students RR folder Same as office code	ENA	4
107. Follow_Up_Date	79	Date of follow up on a periodical subscription	291082	6
108. Follow_Up_Remarks	79	Comments about following up a periodical subscription	1ST FOLLOW UP ON 290482	50
109. Foreign_Service_ Ind	74	Indicates whether a given student is a foreign student	" " (non-foreign); F (foreign)	1
110. Funct_Acct_Code	22	Functional Account Code (refinement of AFSC)	4416	4
111. Function	65	Free text field describing the use of a room	CLASSROOM	25
112. Funding	36	Indicates funding source for a foreign student	MA (military assist.)	2
113. Fund_Type	50*	Indicates the type of funding for a given MMEP tuple	FM (funded military)	2
114. GMAT	31,74,76	Total GMAT score for a given student	1234	4
115. GMAT_Ind	87	Indicates whether a student has GMAT scores	Y (yes); N (no)	1
116. Grade	37,53,83	Letter grade a student made in a given course	A-	2
117. Grade_Type	83	Type of special grade	PF (pass/fail)	3
118. Graduation_Date	67	Date a particular section will graduate	171282	6
119. GRE	31,74,76	Total GRE score for a given student	1234	4
120. GRE_Ind	87	Indicates whether a student has GRE scores	Y (yes); N (no)	1
121. Group_Contact_Hrs	32	Total number of hours per week a given faculty member meets with students as a group	16.0	4
122. HPhone	71,74	Home phone number	5132533936	10
123. ID_No	13*,36*	For relation 13 it is the SSAN For relation 36 it is the foreign students unique ID number	999999999	9
124. Ind_Contact_Hrs	32	Total number of hours per week a faculty member meets with individual students	10.0	4
125. Initial_Analysis	30	Free text field for indicating the initial analysis of a faculty board about what circumstances in a givens students situation lead to the need for a faculty board	POOR ACADEMIC BACKGROUND	50

Appendix I,F

126. Instructor_No	32	Indicates which instructor a given faculty member is in a multi-instructor course	2 (second instr.)	1
127. Lab_Hrs	3	Total number of laboratory hours associated with a given course	4	1
128. Last Organization	63	Free text field indicating the name of a students last organization	6964th COMPUTER SERVSQ (ELC)	30
129. Leader_SSAN	67	SSAN of the section leader of a given section	222334444	9
130. Lec Hrs	3	Total number of lecture hours associated with a given course	4	1
131. Legal Residence	14	Free text field indicating the address of a students legal residence	3790 PARK SOUTH TOPEKA, KS. 66601	50
132. Letter Color	9	Color of the letters on a library binding	COL (gold)	3
133. Local Address	41,71,74,76	Free text field indicating a persons local address	2702 KICKAPOO STR. LEAVENWORTH, KS. 74321	50
134. Location	35*,56,60,82	Free text field indicating the name of a location	ACM CONFERENCE	30
135. Location_Code	18*,39*,40*,41*,47*,48*,49*,50*,54*,66*,88*	Code for up to 99,999 locations. Keys to a unique location name and location address	98765	5
136. Location_Name	41	Complete english name corresponding to a unique location code	EDWARDS AIR FORCE BASE	30
137. Locker_No	42*	Number of a students locker	231A	4
138. MAJCOM	44*,54*	Short version of major command. Keys to unique long version	USAFE	5
139. MAJCOM_Title	44	Unique long english version of major command	US AIR FORCES IN EUROPE	30
140. Major	21	Free text field indicating english version of a students major or primary sequence	INFORMATION SYSTEMS	25
141. Manning_Code	87	Indicates student type	USE UNCLEAR	2
142. Marital_Status	71,74,76	Indicates whether a person is married	M (married); S (single)	1
143. Max_Credit	3	Indicates the max number of credits possible for a variable credit course	4	1
144. Max_Size	64	Indicates the max number of people a room can hold	999	3
145. MCAT_Score	46	Medical College Aptitude Test Total (range 0 to 90)	75	2
146. MCAT_Score_Date	46	MCAT score date	081052	6
147. Med_Program	46	Type of medical program a student is in	MEDICINE/NEUROLOGY	20
148. Med_Spec_Code	45*,47	Medical specialty code (AFSC with suffix)	9326A	6
149. MICRO_Score	45	Microbiology test score on medical board certification exam	500	3
150. Military_Spouse	70	Indicates whether a given spouse is in the military	M (military spouse); " " (nonmilitary spouse)	1
151. Min_Credit	3	Indicates the minimum number of credits possible for a variable credit course	2	1
152. Min_Enrollment	84	Minimum enrollment required for a course	5	2
153. Minor	21	English version of a students minor or secondary sequence	SOFTWARE ENGINEERING	25
154. Month	2*,9,65*	Short english abbreviation for a month	DEC	3
155. Name	34,51,69,70,71,74,76	Name of an individual	JEFFREY S. RICKS	30
156. No_Added	84	Total number of students who added a course after initial registration	10	2
157. No_Audits	84	Total number of students who have registered to audit a class	2	2

158. No_Avail	10	Indicates the total number of a specific book available at the bookstore	50	3
159. No-Deps	70	Total number of dependents other than spouse	3	2
160. No_Enrolled	84	Total number of students in a class	35	2
161. No_Not_ Registered	84	Total number of students enrolled without formal registration including waivers, exemptions, advanced standing and equivalency	40	2
162. No_Ordered	11	Total number of a specific book on a bookstore order	100	3
163. No_Required	16	Total number of a specific course book required for a particular course	20	2
164. No_Students	50,54,67	Total number of students enrolled in a particular course	13	3
165. No_Non_AF	67	Total number of non-Air Force students in a section	10	2
166. Office	10,27,34, 38*,52*,71	Short version of a unique office symbol. Keys to a unique longer version	ENA	4
167. Office_Title	52	Unique longer version of an office title	ENGINEERING ADMIN	30
168. Old_Code	88	Previous student code (e.g. resident, CI, or transfer) for transfer students	CI	2
169. Order_No	11*	Purchase order number for a bookstore book	490657	6
170. Org_Abbrev	57*,58*	Unique short version of a professional organization name. Keys to a unique longer version	IEEE	5
171. Organization	51,56*,69	Long name of a organization	AF LOGISTICS COMMAND	25
172. Org_Name	57	Unique long version of an organization also having a shorter org_abbrev	INSTITUTE OF ELECTRICAL & ELECTRONIC ENGINEERS	50
173. Output_APSC	14	APSC a student leaves an AFIT program with	5135A	6
174. Output_MAJCOM	87	Unique abbrev. for major command a student 1st goes to from AFIT	SAC	5
175. PAFSC	71,74,76	Permanent Air Force Specialty code	5131B	6
176. PATH_Score	45	Pathology score on med. board certification exam	500	3
177. Patron Name	13	Name of a person who checked out material from the AFIT library	ROBERT S. COLBURN	30

178. Payment_Type	40*,77*	Indicates the type of payment	TU (tuition);
			BK (book);
			FE (fee);
			TH (thesis)
				4

179. Per_Diem	80	Amount of money paid for per diem	9999.99	7
180. PHARM_Score	45	Pharmacology score of med board certification test	500	3
181. Photo_Date	71	Date of most recent photo for a staff member	311251	6
182. PHYS_Score	45	Physiology score on med board certification test	500	3
183. POC	2,18,39	Point of contact	DR. MORTIMER SNERD	30
184. Pos_No	34*,71,73*	Air Force position number	0104010	7
185. Prerequisite	55*	Indicates the course number of a course which is the prerequisite for another course	EE758	6
186. Prev_GPA	87	Cumulative GPA of transfer student at time of transfer	3.25	4
187. Previous_MAJCOM	71,74	Unique abbreviation for previous major command of a person	SAC	5
188. Price	10,27,79	Price of an item	35.49	7
189. Primary_Ed_Level	33	Primary degree of a faculty member	PHD, CONSTRUCTION	30

			ENGINEERING	
190. Prior_AFIT	14,87	Total number of months a student attended AFIT before this assignment	18	2
191. Prior_GPA	14	Prior cumulative GPA of an AFIT CI student	3.10	4
192. Prior_Grade	83	Grade a student previously had in a particular course he/she took before	C-	2
193. Producer	7*	Name of producer of a particular Audio-Visual resource	CECIL B. DEMILL	30
194. Production_Date	7	Date a particular Audio-Visual resource was produced	080933	6
195. Program	14,15*,18*	Code for AFIT programs (5 letter ed code)	POCYY	5
196. Proj_Enrollment	84	Projected enrollment in a course	55	2
197. Pub_Kind	60	Indicates a partirular type of publication	MONOGRAPH	15
198. Publisher	10	Name and address of a publisher of a particular book	RANDOM HOUSE NEW YORK, NEW YORK	50
199. Qual_Pts	83	Quality points earned by a student	500	5
200. Quantitative	86	Quantitative points scored on a GRE test	500	3

201. Quarter	16*,17*,26*,
	30*,31*,32*,
	37*,53*,88
		Abbreviation for quarter	SP (spring);
			SU (summer);
			FA (fall); WI(winter)
				2

202. Quota_Type	61*	Type of program quota	E (estimate); A (actual)	1
203. Rank	34,35*,69, 71,74,76	Rank or Grade abbrev	CAPT; GS12	4
204. Re_Admit_Term	87	Quarter a student was readmitted	SP (spring)	2
205. Re_Admit_Year	87	Year a student was readmitted	82	2
206. Reason	30	Reason faculty board was called	GPA < 2.5	30
207. Reader1	19	SSAN of the 1st thesis reader	444556666	9
208. Reader2	19	SSAN of the 2nd thesis reader	555667777	9
209. Registered_Hrs	83	Total number of hours a student has registered in while at AFIT	80	3
210. Remarks	2,3,25,30	Free text field for remarks or comments	THIS PLACE SUCKS	50
211. Renewal_Date	79	Date subscription ends and should be renewed	121282	6
212. Requester	79	Name of a person requesting a subscription	DR. HENRY POTOCZNY	30
213. Return_Date	80	Date a person returns from TDY	130282	6
214. Room	2*,17,49*, 64*,65,68*,71	Room number	234	4
215. Score_Date	45	Date of med board certification	140382	6
216. Secondary_Ed_Level	33	Secondary degree of a faculty member	MS, MATHEMATICS	30
217. Second_Author	81	SSAN of the 2nd author of the thesis	666778888	9
218. Section	67*,74	Abbrev for a resident section course section	GCS82D	6
219. Section_Control	3	A code indicating special processing rules for course sections	R (regular control)	1

220. Security Clearance	71,74	Abbrev for security clearance levels	TS (top secret);
			S (secret);
			CF (confidential);
			UN (unclassified)
				2

221. Service	36,71,74,76	Abbrev for branch of service	USAF; USAFR	5
222. Servicing_AFO	66	Location code for servicing Air Force organization	23456	5
223. Sex	71,74,76	Indicates whether a person is male or female	M (male); F (female)	1
224. Source	60*	Free text field indicating the name of the journal a given article was published in	ACM PROCEEDINGS	30

Appendix I,F

225. Speaker	7	Free text field indicating the name of the narrator or main speaker on an audio-visual resource	ROD STERLING	30
226. Special_Grading	3	Indicates type of grading for a particular course	P (pass/fail only)	1
227. Special_Interest	33	Free text field indicating any special interests a faculty member might have	ARTIFICIAL INTELLIGENCE	30
228. Special_Room_Type	3,64	Indicates special attributes needed or existent in a room	A (auditorium); L (lab)	1
229. Sponsors_SSAN	70*	SSAN of a spouses husband or wife	111223333	9
230. Spouse_Type	70	Indicates whether a given spouse tuple is for a staff member or student	S (staff); " " (student)	1
231. SSAN	(see note 4)	Social security account number	444560937	9
232. Starting_Time	2*,17*,65*,68	Starting time for a scheduled event (24 hr clock)	0800	4
233. Status	11,14,34	Free text field indicating status	BOOK OVERDUE	30
234. Stock No	27*	Stock number for a particular piece of equipment	A7691534	13
235. Student Limit	3,54	Maximum number of students allowed to register for a particular course	40	2
236. Subject	7,56,60,81	Free text field describing a topic or subject	ACADEMIC RAPE	40
237. Success	30	Free text field indicating how well the student was able to carry out the recommendations of a faculty board	STUDENT SUCCESSFUL	30
238. Supporting INST Code	48	Location code for the institution supporting an MNEP program	54321	5
239. Suspense Action	14	Free text field giving an abbreviated description of the action to be taken on a given suspense	CONTACT STUDENT	30
240. Suspense Date	14	Deadline for a suspense action	180683	6
241. Term GPA	14,31	Grade point average for a particular student for a particular term	2.50	4
242. Thesis No	19,81*	Unique number for a given students thesis	AFITCCSEE880D-3	15
243. Thesis Pay	14	Indicates whether a given student has been sent his thesis pay	Y (yes); N (no)	1
244. Thesis Pub Date	81	Date an individual thesis was published	101263	6
245. Thesis Sponsor	81,82*	Name of the sponsor of an individual thesis	AFIT/CI	30
246. Time	7	Length of an audio-visual resource in minutes	120	3
247. Title	(see note 5)	Full title of an item	DATABASE MADE EASY	30
248. TMST	71,74	Date total military service time figured from	151075	6
249. Total	85,86	Total score on GRE/GMAT tests	900	3
250. Transfer Ind	87	Indicates whether a student is a transfer student	T (transfer); " " (non-transfer)	1
251. Travel	80	Amount of TDY money spent on travel	999.99	7
252. TRF Hrs	88	Total earned hours awarded for the transfer activity	160	4
253. TRF In Date	88	Date transfer work began	81053	6
254. TRF End Date	88	Date transfer work ended	81053	6
255. TRF Qual Hrs	88	Total transfer quality hours for the transfer activity	300	4
256. TRF Qual Pts	88	Total transfer quality points for the transfer activity	250	4
257. Trouble Term	30	Term a student had academic trouble	SU (summer)	2
258. Trouble Year	30	Year a student had academic trouble	82	2
259. Tuition_Rate	66	Rate an institution charges AFIT for tuition	95.00	7
260. UG_CGPA	31,87	Undergraduate cumulative GPA	3.01	4
261. USAF	59	Total number of eligible AF personnel promoted from a particular board	1000	4
262. USAF_Elig	59	Total % of eligible AF personnel promoted from a particular board	95.2	4
263. Verbal	86	Verbal scores on a particular students GRE test	500	3

264. Volume	9	Volume number of a bound periodical title	117B	4
265. Wait_List_No	17,84	Total number of students waiting for a particular course and/or course time	14	2
266. Year	(see note 6)	Last two digits of a calendar year	82	2
267. YR1	61	1st year quota value in a quota tuple	100	3
26R. YR2	61	2nd year quota value in a quota tuple	100	3
269. YR3	61	3rd year quota value in a quota tuple	100	3
270. YR4	61	4th year quota value in a quota tuple	100	3
271. YR5	61	5th year quota value in a quota tuple	100	3

NOTES

1. Any attribute which has a starred relation number in the RELATION column, is either a key or a part of a key in the starred relation.
2. All dates are in the format (day, month, year), e.g., 171282.
3. " " indicates a blank
4. Relations that SSAN is in are: 5*,8*,12*,14*,19*,21*,22*,23*,25*,26*,29*,30*,31*,32*,33*,35*,37*,42*,45*,46*,47*,53*,
56*,58*,60*,63*,69*,71*,72*,73,74*,75*,76*,77*,78*,80*,83*,85*,86*,87*,88*
5. Relations that Title is in are: 3,7*,9*,10*,11,16*,38*,43,56,60*,68,79*,81
6. Relations that Year is in are: 2*,9,16*,17*,26*,30*,31*,32*,37*,53*,65*

LOGICAL DATABASE VIEWS, RELATIONAL LEVEL FOR CADIS, BY OFFICE

LOGICAL DATABASE VIEW FOR:

SCHOOL OF CIVIL ENGINEERING (AFIT/DE)

1. ACADEMIC TITLES:
Academic Title Code*, Academic_Title.

2. AFIT_CALENDAR:
Day*, Month*, Year*, Starting_Time*, Activity, CC_Activity, Room*, Building*, POC, Remarks.

3. AFIT_COURSES:
Course*, Title, Description, Min_Credit, Max_Credit, Special_Grading, Section_Control, Remarks, Lec_Hrs, Lab_Hrs, Student_Limit, Special_Room_Type.

4. AFSC:
AFSC*, Description.

20. DEGREES:
Degree_Code*, Description.

22. DE_DATA:
SSAN*, DAFSC, Funct_Acct_Code, DOS, Arrival_Date.

24. ED_CODES:
Ed_Code*, Ed_Code_Title.

25. ED_HIST:
SSAN*, College*, Begin Date, End_Date, Degree_Code, Departure Reason, Remarks.

33. FAC ED:
SSAN*, Primary_Ed_Level, Secondary_Ed_Level, Special Interest.

36. FOREIGN_STUDENTS_DATA:
ID No*, Service, Country, Funding.

41. LOCATIONS:
Location Code*, Location Name, Local Address.

44. MAJCOMS:

MAJCOM*, MAJCOM Title.

52. OFFICES:
Office*, Office Title.

53. PAST COURSES:
SSAN*, College*, Course*, Grade, Quarter*, Year*.

70. SPOUSES:
Sponsors SSAN*, Name, DOB, No Deps, Spouse Type, Military Spouse.

71. STAFF:
SSAN*, Name, Rank, HPhone, DPhone, Room, Office, Pos No, Duty Title, Date Assigned, Departure Date, Local_Address, DOB, DOR, Acad_Title_Code, DOAR, Marital_Status, Sex, PAFSC, DAFSC, Ethnic_Group, Service, Photo_Date, Security_Clearance, TMST, Previous_MAJCOM.

72. STAFF_DE_DATA:
SSAN*, Course_Directed*.

74. STUDENTS:
SSAN*, Name, Rank, PAFSC, Ed_Code, DOR, Sex, DOB, Advisors_SSAN, Entry_Date, DPhone, HPhone, Local_Address, Foreign_Service_Ind, Service, Departure_Date, Security_Clearance, Duty_Location, Aero_Rating, Previous_MAJCOM, Marital_Status, TMST, Ethnic_Group, CRE, GMAT, Duty_Title, Section, Academic_Standing.

LOGICAL DATABASE VIEW FOR:

CIVILIAN INSTITUTION PROGRAMS (AFIT/CI)

2. AFIT_CALENDAR:
Day*, Month*, Year*, Starting_Time*, Activity, CC_Activity, Room*, Building*, POC, Remarks.

4. AFSC:
AFSC*, Description.

6. ALLOWANCE/PAYMENT_TYPES:
Allowance_Type*, Description.

14. CI_DATA:
SSAN*, Program, Output_AFSC, Degree_Code, Prior_GPA, Available_Date, Prior_AFIT, Departure_Reason, Suspense_Date, Legal_Residence, Thesis_Pay, Suspense_Action, Status, Begin_Date, C_GPA, Term_GPA.

15. CI_PROGRAMS:
Program*, Description.

18. CURRENT_CI_PROGRAMS:
Location_Code*, Program*, POC, DPhone.

20. DEGREES:
Degree_Code*, Description.

21. DEGREE_SOUGHT:
SSAN*, Degree_Code, Major, Minor.

24. ED_CODES:
Ed_Code*, Ed_Code_Title.

25. ED_HIST:
SSAN*, College*, Begin_Date, End_Date, Degree_Code, Departure_Reason, Remarks.

26. ED_PLANS:
SSAN*, Course*, Quarter*, Year*, Credits.

36. FOREIGN_STUDENTS_DATA:
ID_No*, Service, Country, Funding.

37. GRADES:
SSAN*, Course*, Quarter*, Year*, Grade.

39. INSTITUTION_POCS:
Location_Code*, POC, DPhone.

40. INST_PAYMENTS:
Location_Code*, Date*, Payment_Type*, Amount.

41. LOCATIONS:
Location_Code*, Location_Name, Local_Address.

44. MAJCOMS:
MAJCOM*, MAJCOM_Title.

45. MED_BOARD_CERTIFICATION:
SSAN*, Med_Spec_Code*, Certification_Date, ANAT_Score, PHYS_Score, BIOCH_Score, PATH_Score, MICRO_Score, PHARM_Score, BEHSCI_Score, Score_Date.

46. MED_PROGRAM:
SSAN*, MCAT_Score, MCAT_Score_Date, Degree_Code, Med_Program, Accession_Source.

47. MED_TOURS:
SSAN*, Med_Spec_Code, Location_Code*, Begin_Date*, End_Date.

48. MMEP:
Location_Code*, Supporting_Inst_Code.

49. MMEP_CLASSROOMS:
Location_Code*, Room*, Building*.

50. MMEP_FUNDING:
Location_Code*, Fund_Type*, No_Students.

53. PAST_COURSES:
SSAN*, College*, Course*, Grade, Quarter*, Year*.

54. PCE:
Course*, Location_Code*, MAJCOM*, Student_Limit, No_Students.

66. SCHOOL/EWI_DATA:
Location_Code*, Servicing_AFO, Tuition_Rate.

70. SPOUSES:
Sponsors_SSAN*, Name, DOB, No_Deps, Spouse_Type, Military_Spouse.

74. STUDENTS:
SSAN*, Name, Rank, PAFSC, Ed_Code, DOR, Sex, DOB, Advisors_SSAN, Entry_Date, DPhone, HPhone, Local_Address, Foreign_Service_Ind, Service, Departure_Date, Security_Clearance, Duty_Location, Aero_Rating, Previous_MAJCOM, Marital_Status, TMST, Ethnic_Group, GRE, GMAT, Duty_Title, Section, Academic_Standing.

77. STUDENT_PAYMENTS:
SSAN*, ESA_No*, Date*, Payment_Type*, Amount.

78. STUDENT_SCHED:
SSAN*, Course*.

LOGICAL DATABASE VIEW FOR:

ACADEMIC LIBRARY (AFIT/LD)

1. ACADEMIC_TITLES:
   Academic_Title_Code*, Academic_Title.

2. AFIT_CALENDAR:
   Day*, Month*, Year*, Starting_Time*, Activity, CC_Activity, Room*, Building*, POC, Remarks.

3. AFIT_COURSES:
   Course*, Title, Description, Min_Credit, Max_Credit, Special_Grading, Section_Control, Remarks, Lec_Hrs, Lab_Hrs, Student_Limit, Special_Room_Type.

4. AFSC:
   AFSC*, Description.

7. AUDIO_VIS:
   Title*, Time, Producer*, Speaker, Subject, Production_Date.

9. BINDINGS:
   Title*, Volume, Month, Year, Color_No, Letter_Color.

11. BOOK_ORDERS:
    Author, Title, Order_No*, No_Ordered, Date, Status.

12. BOXES:
    SSAN*, Box_No*.

13. CHECKOUTS:
    Call_No*, Patron_Name, ID_No*, Due_Date.

17. COURSE_TIMES:
    Course*, Starting_Time*, Day*, Quarter*, Year*, Room, Building, Course_Status, Wait_List_No, Begin_Date, End_Date.

24. ED_CODES:
    Ed_Code*, Ed_Code_Title.

36. FOREIGN_STUDENTS_DATA:
    ID_No*, Service, Country, Funding.

38. HOLDINGS:
    Title*, Office*, Begin_Date, End_Date.

41. LOCATIONS:
    Location_Code*, Location_Name, Local_Address.

43. LIBRARY_BOOKS:
Call_No*, Title, Author.

44. MAJCOMS:
MAJCOM*, MAJCOM_Title.

51. NON_STUDENTS:
Name, Card_No*, Organization, DPhone.

52. OFFICES:
Office*, Office_Title.

60. PUBLICATIONS:
SSAN*, Title*, Date, Subject, Source*, Location, Pub_Kind, Co_Auth_Name.

62. RESERVES:
Call_No*, FAC_SSAN*, Date_Reserved*.

68. SHORT_COURSE:
Course*, Title, Begin_Date*, End_Date, Room*, Building*, Starting_Time, Description.

69. SHORT_STUDENTS:
SSAN*, Course*, Begin_Date*, Name, Rank, Organization, DPhone

71. STAFF:
SSAN*, Name, Rank, HPhone, DPhone, Room, Office, Pos_No, Duty_Title, Date_Assigned, Departure_Date, Local_Address, DOB, DOR, Acad_Title_Code, DOAR, Marital_Status, Sex, PAFSC, DAFSC, Ethnic_Group, Service, Photo_Date, Security_Clearance, TMST, Previous_MAJCOM.

74. STUDENTS:
SSAN*, Name, Rank, PAFSC, Ed_Code, DOR, Sex, DOB, Advisors_SSAN, Entry_Date, DPhone, HPhone, Local_Address, Foreign_Service_Ind, Service, Departure_Date, Security_Clearance, Duty_Location, Aero_Rating, Previous_MAJCOM, Marital_Status, TMST, Ethnic_Group, GRE, GMAT, Duty_Title, Section, Academic_Standing.

79. SUBSCRIPTIONS:
Title*, Check_In, Renewal_Date, Price, Expiration_Date, Requester, Begin_Date, Follow_Up_Date, Follow_Up_Remarks.

81. THESES:
Thesis_No*, Title, Thesis_Sponsor, Classification, First_Author, Second_Author, Subject, Thesis_Pub_Date, DTIC_No, Cleared_For_Release.

82. THESIS_SPONSORS:
    Thesis_Sponsor*, Location.

LOGICAL DATABASE VIEW FOR:

DIRECTOR OF ACADEMIC AFFAIRS (AFIT/CAE)

1. ACADEMIC_TITLES:
   Academic_Title_Code*, Academic_Title.

2. AFIT_CALENDAR:
   Day*, Month*, Year*, Starting_Time*, Activity, CC_Activity, Room*, Building*, POC, Remarks.

3. AFIT_COURSES:
   Course*, Title, Description, Min_Credit, Max Credit, Special Grading, Section_Control, Remarks, Lec Hrs, Lab Hrs, Student Limit, Special Room Type.

4. AFSC:
   AFSC*, Description.

5. ALLOWANCE_PAYMENTS:
   SSAN*, Allowance_Type*, Date_Paid*, Amount.

6. ALLOWANCE/PAYMENT_TYPES:
   Allowance_Type*, Description.

7. AUDIO_VIS:
   Title*, Time, Producer*, Speaker, Subject, Production_Date.

8. AWARDS:
   SSAN*, Award, Date*.

9. BINDINGS:
   Title*, Volume, Month, Year, Color_No, Letter_Color.

10. BOOKS:
    Title*, Author*, Publisher, No_Avail, Price, Office.

11. BOOK_ORDERS:
    Author, Title, Order_No*, No_Ordered, Date, Status.

12. BOXES:
    SSAN*, Box_No*.

13. CHECKOUTS:
    Call_No*, Patron_Name, ID_No*, Due_Date.

14. CI_DATA:
    SSAN*, Program, Output_AFSC, Degree_Code, Prior_GPA, Available_Date, Prior_AFIT, Departure_Reason, Suspense_Date, Legal_Residence, Thesis_Pay,

Suspense Action, Status, Begin Date, C_GPA, Term_GPA.

15. CI_PROGRAMS:
    Program*, Description.

16. COURSE_BOOKS:
    Course*, Author*, Title*, No_Required, Quarter*, Year*.

17. COURSE_TIMES:
    Course*, Starting_Time*, Day*, Quarter*, Year*, Room, Building, Course_Status, Wait_List_No, Begin_Date, End_Date.

18. CURRENT_CI_PROGRAMS:
    Location_Code*, Program*, POC, DPhone.

19. CURRENT_THESES:
    SSAN*, Thesis_No, Advisors_SSAN, Reader1, Reader2.

20. DEGREES:
    Degree_Code*, Description.

21. DEGREE_SOUGHT:
    SSAN*, Degree_Code, Major, Minor.

22. DE_DATA:
    SSAN*, DAFSC, Funct_Acct_Code, DOS, Arrival_Date.

23. DUTY_OFFICER:
    SSAN*, Duty, Credit_Date, Duty_Date.

24. ED_CODES:
    Ed_Code*, Ed_Code_Title.

25. ED_HIST:
    SSAN*, College*, Begin_Date, End_Date, Degree_Code, Departure_Reason, Remarks.

26. ED_PLANS:
    SSAN*, Course*, Quarter*, Year*, Credits.

27. EQUIPMENT:
    Stock_No*, Description, Price, Date_Ordered*, Date_Received, Availability, Office.

28. EXTRA_DUTIES:
    Extra_Duty_Code*, Extra_Duty_Title.

29. EXTRA_DUTY_ROSTER:
    SSAN*, Extra_Duty_Code*, Date.

30. FAC_BOARD:
SSAN*, Trouble_Term, Quarter*, Year*, Trouble_Year, Reason, Initial_Analysis, Board_Action, Success, Remarks.

31. FAC_BOARD_STUD_DATA:
SSAN*, Quarter*, Year*, UG_CGPA, GRE, GMAT, Term_GPA, C_GPA, A_Cred, B_Cred, C_Cred.

32. FAC_COURSES:
SSAN*, Course*, Quarter*, Year*, Instructor_No, Group_Contact_Hrs, Ind_Contact_Hrs.

33. FAC_ED:
SSAN*, Primary_Ed_Level, Secondary_Ed_Level, Special_Interest.

34. FAC_REPLACEMENT:
Pos_No*, Office, DAFSC, Rank, Name, Arrival_Date, Degree_Required, Academic_Specialty, Status, DPhone.

35. FAC_WORK:
SSAN*, Location*, Rank*, Begin_Date, End_Date.

36. FOREIGN_STUDENTS_DATA:
ID_No*, Service, Country, Funding.

37. GRADES:
SSAN*, Course*, Quarter*, Year*, Grade.

38. HOLDINGS:
Title*, Office*, Begin_Date, End_Date.

39. INSTITUTION_POCS:
Location_Code*, POC, DPhone.

40. INST_PAYMENTS:
Location_Code*, Date*, Payment_Type*, Amount.

41. LOCATIONS:
Location_Code*, Location_Name, Local_Address.

42. LOCKERS:
SSAN*, Locker_No*.

43. LIBRARY_BOOKS:
Call_No*, Title, Author.

44. MAJCOMS:
MAJCOM*, MAJCOM_Title.

45. MED_BOARD_CERTIFICATION:
SSAN*, Med_Spec_Code*, Certification_Date, ANAT_Score,

PHYS_Score, BIOCH_Score, PATH_Score, MICRO_Score, PHARM_Score, BEHSCI_Score, Score_Date.

46. MED_PROGRAM:
SSAN*, MCAT_Score, MCAT_Score_Date, Degree_Code, Med_Program, Accession_Source.

47. MED_TOURS:
SSAN*, Med_Spec_Code, Location_Code*, Begin_Date*, End_Date.

48. MMEP:
Location_Code*, Supporting_Inst_Code.

49. MMEP_CLASSROOMS:
Location_Code*, Room*, Building*.

50. MMEP_FUNDING:
Location_Code*, Fund_Type*, No_Students.

51. NON_STUDENTS:
Name, Card_No*, Organization, DPhone.

52. OFFICES:
Office*, Office_Title.

53. PAST_COURSES:
SSAN*, College*, Course*, Grade, Quarter*, Year*.

54. PCE:
Course*, Location_Code*, MAJCOM*, Student_Limit, No_Students.

55. PREREQUISITES:
Course*, Prerequisite*.

56. PRESENTATIONS:
SSAN*, Date*, Title, Location, Organization*, Subject.

57. PROFESSIONAL_ORGS:
Org_Abbrev*, Org_Name.

58. PROF_FAC_ORGS:
SSAN*, Org_Abbrev*.

59. PROMOTION_STATS:
Board_Date*, Board_Kind*, USAF, AFIT, USAF_Elig, AFIT_Elig.

60. PUBLICATIONS:
SSAN*, Title*, Date, Subject, Source*, Location, Pub_Kind, Co_Auth_Name.

61. QUOTAS:
Ed_Code*, Quota_Type*, YR1, YR2, YR3, YR4, YR5.

62. RESERVES:
Call_No*, FAC_SSAN*, Date_Reserved*.

63. RESIDENT_DATA:
SSAN*, Last_Organization, Duration, DOC, Emergency_Address.

64. ROOMS:
Room*, Building*, Max_Size, Special_Room_Type.

65. ROOM_SCHEDULE:
Day*, Month*, Year*, Starting_Time*, Ending_Time, Room, Building*, Function.

66. SCHOOL/EWI_DATA:
Location_Code*, Servicing_AFO, Tuition_Rate.

67. SECTION:
Section*, Graduation_Date, Entry_Date, No_Students, Leader_SSAN, Advisors_SSAN, Non_AF.

68. SHORT_COURSE:
Course*, Title, Begin_Date*, End_Date, Room*, Building*, Starting_Time, Description.

69. SHORT_STUDENTS:
SSAN*, Course*, Begin_Date*, Name, Rank, Organization, DPhone

70. SPOUSES:
Sponsors_SSAN*, Name, DOB, No_Deps, Spouse_Type, Military_Spouse.

71. STAFF:
SSAN*, Name, Rank, HPhone, DPhone, Room, Office, Pos_No, Duty_Title, Date_Assigned, Departure_Date, Local_Address, DOB, DOR, Acad_Title_Code, DOAR, Marital_Status, Sex, PAFSC, DAFSC, Ethnic_Group, Service, Photo_Date, Security_Clearance, TMST, Previous_MAJCOM.

72. STAFF_DE_DATA:
SSAN*, Course_Directed*.

73. STAFF_POSITION_DATA:
Pos_No*, SSAN, Ed_Code_Required, Ed_Code_Assigned, Degree_Level_Required, Degree_Level_Assigned.

74. STUDENTS:
SSAN*, Name, Rank, PAFSC, Ed_Code, DOR, Sex, DOB,

Advisors_SSAN, Entry_Date, DPhone, HPhone, Local_Address, Foreign_Service_Ind, Service, Departure_Date, Security_Clearance, Duty_Location, Aero_Rating, Previous_MAJCOM, Marital_Status, TMST, Ethnic_Group, GRE, GMAT, Duty_Title, Section, Academic_Standing.

75. STUDENT_AWARDS:
SSAN*, Award, Date*.

76. STUDENT_HIST:
SSAN*, Name, Rank, PAFSC, Ed_Code, Sex, Entry_Date, Departure_Date, Service, Local_Address, Address_Date, Aero_Rating, Marital_Status, Ethnic_Group, GRE, GMAT, Degree_Code.

77. STUDENT_PAYMENTS:
SSAN*, ESA_No*, Date*, Payment_Type*, Amount.

78. STUDENT_SCHED:
SSAN*, Course*.

79. SUBSCRIPTIONS:
Title*, Check_In, Renewal_Date, Price, Expiration_Date, Requester, Begin_Date, Follow_Up_Date, Follow_Up_Remarks.

80. TDY:
Destination*, SSAN*, Departure_Date*, Return_Date, Estimated_Cost, Actual_Cost,Fees, Per_Diem, Travel.

81. THESES:
Thesis_No*, Title, Thesis_Sponsor, Classification, First_Author, Second_Author, Subject, Thesis_Pub_Date, DTIC_No, Cleared_For_Release.

82. THESIS_SPONSORS:
Thesis_Sponsor*, Location.

RELATIONS PERTAINING TO THE REGISTRARS INFORMATION

83. COURSE_GRADES:
SSAN*, Course*, Grade_Type, Grade, Prior_Grade, Registered_Hrs, Earned_Hrs, Qual_Pts, Drop_Reason, Date.

84. COURSE_STATS:
Course*, Proj_Enrollment, Min_Enrollment, No_Enrolled, Demand, Drops, Wait_List_No, No_Not_Registered,

No_Added, No_Audits.

85. GMAT_SCORES:
SSAN*, Total.

86. GRE_SCORES:
SSAN*, Verbal, Quantitative, Total.

87. RR DATA:
SSAN*, Manning Code, Output_MAJCOM, UG_CGPA, Degree_Code, Prior_AFIT, Folder_Location, Departure_Reason, Admission Action, Transfer Ind, Prev GPA, C GPA, GRE Ind, GMAT Ind, Re Admit Term, Academic Elig, Calc Required Term, Degree Checkout, Admin Hold, Re Admit Year, Calc Required Year.

88. TRANSFERS:
SSAN*, Location Code*, TRF Hrs, TRF Qual Hrs, TRF Qual Pts, Begin_Date, Quarter, TRF_In Date, TRF End Date, Old_Code.

LOGICAL DATABASE VIEW FOR:

COMMAND SECTION, COMMANDANT (AFIT/CC)

1. ACADEMIC_TITLES:
   Academic_Title_Code*, Academic_Title.

2. AFIT_CALENDAR:
   Day*, Month*, Year*, Starting_Time*, Activity, CC_Activity, Room*, Building*, POC, Remarks.

3. AFIT_COURSES:
   Course*, Title, Description, Min_Credit, Max_Credit, Special_Grading, Section_Control, Remarks, Lec_Hrs, Lab_Hrs, Student_Limit, Special_Room_Type.

4. AFSC:
   AFSC*, Description.

5. ALLOWANCE_PAYMENTS:
   SSAN*, Allowance_Type*, Date_Paid*, Amount.

6. ALLOWANCE/PAYMENT_TYPES:
   Allowance_Type*, Description.

7. AUDIO_VIS:
   Title*, Time, Producer*, Speaker, Subject, Production_Date.

8. AWARDS:
   SSAN*, Award, Date*.

9. BINDINGS:
   Title*, Volume, Month, Year, Color_No, Letter_Color.

10. BOOKS:
    Title*, Author*, Publisher, No_Avail, Price, Office.

11. BOOK_ORDERS:
    Author, Title, Order_No*, No_Ordered, Date, Status.

12. BOXES:
    SSAN*, Box_No*.

13. CHECKOUTS:
    Call_No*, Patron_Name, ID_No*, Due_Date.

14. CI_DATA:
    SSAN*, Program, Output_AFSC, Degree_Code, Prior_GPA, Available_Date, Prior_AFIT, Departure_Reason, Suspense_Date, Legal_Residence, Thesis_Pay,

Suspense_Action, Status, Begin Date, C_GPA, Term_GPA.

15. CI_PROGRAMS:
Program*, Description.

16. COURSE_BOOKS:
Course*, Author*, Title*, No_Required, Quarter*, Year*.

17. COURSE_TIMES:
Course*, Starting_Time*, Day*, Quarter*, Year*, Room, Building, Course_Status, Wait_List_No, Begin_Date, End_Date.

18. CURRENT_CI_PROGRAMS:
Location_Code*, Program*, POC, DPhone.

19. CURRENT_THESES:
SSAN*, Thesis_No, Advisors_SSAN, Reader1, Reader2.

20. DEGREES:
Degree_Code*, Description.

21. DEGREE_SOUGHT:
SSAN*, Degree_Code, Major, Minor.

22. DE_DATA:
SSAN*, DAFSC, Funct_Acct_Code, DOS, Arrival_Date.

23. DUTY_OFFICER:
SSAN*, Duty, Credit_Date, Duty_Date.

24. ED_CODES:
Ed_Code*, Ed_Code_Title.

25. ED_HIST:
SSAN*, College*, Begin_Date, End_Date, Degree_Code, Departure_Reason, Remarks.

26. ED_PLANS:
SSAN*, Course*, Quarter*, Year*, Credits.

27. EQUIPMENT:
Stock_No*, Description, Price, Date_Ordered*, Date_Received, Availability, Office.

28. EXTRA_DUTIES:
Extra_Duty_Code*, Extra_Duty_Title.

29. EXTRA_DUTY_ROSTER:
SSAN*, Extra_Duty_Code*, Date.

30. FAC_BOARD:
SSAN*, Trouble_Term, Quarter*, Year*, Trouble_Year, Reason, Initial_Analysis, Board_Action, Success, Remarks.

31. FAC_BOARD_STUD_DATA:
SSAN*, Quarter*, Year*, UG_CGPA, GRE, GMAT, Term_GPA, C_GPA, A_Cred, B_Cred, C_Cred.

32. FAC_COURSES:
SSAN*, Course*, Quarter*, Year*, Instructor_No, Group_Contact_Hrs, Ind_Contact_Hrs.

33. FAC_ED:
SSAN*, Primary_Ed_Level, Secondary_Ed_Level, Special_Interest.

34. FAC_REPLACEMENT:
Pos_No*, Office, DAFSC, Rank, Name, Arrival_Date, Degree_Required, Academic_Specialty, Status, DPhone.

35. FAC_WORK:
SSAN*, Location*, Rank*, Begin_Date, End_Date.

36. FOREIGN_STUDENTS_DATA:
ID_No*, Service, Country, Funding.

37. GRADES:
SSAN*, Course*, Quarter*, Year*, Grade.

38. HOLDINGS:
Title*, Office*, Begin_Date, End_Date.

39. INSTITUTION_POCS:
Location_Code*, POC, DPhone.

40. INST_PAYMENTS:
Location_Code*, Date*, Payment Type*, Amount.

41. LOCATIONS:
Location_Code*, Location Name, Local Address.

42. LOCKERS:
SSAN*, Locker No*.

43. LIBRARY BOOKS:
Call No*, Title, Author.

44. MAJCOMS:
MAJCOM*, MAJCOM Title.

45. MED BOARD CERTIFICATION:
SSAN*, Med Spec Code*, Certification Date, ANAT Score,

PHYS Score, BIOCH_Score, PATH_Score, MICRO_Score, PHARM_Score, BEHSCI Score, Score_Date.

46. MED PROGRAM:
SSAN*, MCAT_Score, MCAT_Score_Date, Degree_Code, Med_Program, Accession_Source.

47. MED_TOURS:
SSAN*, Med_Spec_Code, Location_Code*, Begin_Date*, End_Date.

48. MMEP:
Location_Code*, Supporting Inst Code.

49. MMEP CLASSROOMS:
Location_Code*, Room*, Building*.

50. MMEP_FUNDING:
Location_Code*, Fund_Type*, No_Students.

51. NON STUDENTS:
Name, Card_No*, Organization, DPhone.

52. OFFICES:
Office*, Office_Title.

53. PAST_COURSES:
SSAN*, College*, Course*, Grade, Quarter*, Year*.

54. PCE:
Course*, Location_Code*, MAJCOM*, Student_Limit, No_Students.

55. PREREQUISITES:
Course*, Prerequisite*.

56. PRESENTATIONS:
SSAN*, Date*, Title, Location, Organization*, Subject.

57. PROFESSIONAL_ORGS:
Org_Abbrev*, Org_Name.

58. PROF_FAC_ORGS:
SSAN*, Org_Abbrev*.

59. PROMOTION_STATS:
Board_Date*, Board_Kind*, USAF, AFIT, USAF_Elig, AFIT_Elig.

60. PUBLICATIONS:
SSAN*, Title*, Date, Subject, Source*, Location, Pub_Kind, Co_Auth_Name.

61. QUOTAS:
Ed_Code*, Quota_Type*, YR1, YR2, YR3, YR4, YR5.

62. RESERVES:
Call_No*, FAC_SSAN*, Date_Reserved*.

63. RESIDENT_DATA:
SSAN*, Last_Organization, Duration, DOC, Emergency_Address.

64. ROOMS:
Room*, Building*, Max_Size, Special_Room_Type.

65. ROOM_SCHEDULE:
Day*, Month*, Year*, Starting_Time*, Ending_Time, Room, Building*, Function.

66. SCHOOL/EWI_DATA:
Location_Code*, Servicing_AFO, Tuition_Rate.

67. SECTION:
Section*, Graduation_Date, Entry_Date, No_Students, Leader_SSAN, Advisors_SSAN, Non_AF.

68. SHORT_COURSE:
Course*, Title, Begin_Date*, End_Date, Room*, Building*, Starting_Time, Description.

69. SHORT_STUDENTS:
SSAN*, Course*, Begin_Date*, Name, Rank, Organization, DPhone

70. SPOUSES:
Sponsors_SSAN*, Name, DOB, No_Deps, Spouse_Type, Military_Spouse.

71. STAFF:
SSAN*, Name, Rank, HPhone, DPhone, Room, Office, Pos_No, Duty_Title, Date_Assigned, Departure_Date, Local_Address, DOB, DOR, Acad_Title_Code, DOAR, Marital_Status, Sex, PAFSC, DAFSC, Ethnic_Group, Service, Photo_Date, Security_Clearance, TMST, Previous_MAJCOM.

72. STAFF_DE_DATA:
SSAN*, Course_Directed*.

73. STAFF_POSITION_DATA:
Pos_No*, SSAN, Ed_Code_Required, Ed_Code_Assigned, Degree_Level_Required, Degree_Level_Assigned.

74. STUDENTS:
SSAN*, Name, Rank, PAFSC, Ed_Code, DOR, Sex, DOB,

Advisors_SSAN, Entry_Date, DPhone, HPhone, Local_Address, Foreign_Service_Ind, Service, Departure_Date, Security_Clearance, Duty_Location, Aero_Rating, Previous_MAJCOM, Marital_Status, TMST, Ethnic_Group, GRE, GMAT, Duty_Title, Section, Academic_Standing.

75. STUDENT_AWARDS:
SSAN*, Award, Date*.

76. STUDENT_HIST:
SSAN*, Name, Rank, PAFSC, Ed_Code, Sex, Entry_Date, Departure_Date, Service, Local_Address, Address_Date, Aero_Rating, Marital_Status, Ethnic_Group, GRE, GMAT, Degree_Code.

77. STUDENT_PAYMENTS:
SSAN*, ESA_No*, Date*, Payment_Type*, Amount.

78. STUDENT_SCHED:
SSAN*, Course*.

79. SUBSCRIPTIONS:
Title*, Check_In, Renewal_Date, Price, Expiration_Date, Requester, Begin_Date, Follow_Up_Date, Follow_Up_Remarks.

80. TDY:
Destination*, SSAN*, Departure_Date*, Return_Date, Estimated_Cost, Actual_Cost,Fees, Per_Diem, Travel.

81. THESES:
Thesis_No*, Title, Thesis_Sponsor, Classification, First_Author, Second_Author, Subject, Thesis_Pub_Date, DTIC_No, Cleared_For_Release.

82. THESIS_SPONSORS:
Thesis_Sponsor*, Location.

RELATIONS PERTAINING TO THE REGISTRARS INFORMATION

83. COURSE GRADES:
SSAN*, Course*, Grade_Type, Grade, Prior_Grade, Registered_Hrs, Earned_Hrs, Qual_Pts, Drop_Reason, Date.

84. COURSE STATS:
Course*, Proj Enrollment, Min Enrollment, No Enrolled, Demand, Drops, Wait_List_No, No Not Registered,

No_Added, No_Audits.

85. GMAT_SCORES:
SSAN*, Total.

86. GRE SCORES:
SSAN*, Verbal, Quantitative, Total.

87. RR DATA:
SSAN*, Manning_Code, Output_MAJCOM, UG_CGPA, Degree_Code, Prior AFIT, Folder_Location, Departure_Reason, Admission Action, Transfer_Ind, Prev_GPA, C_GPA, GRE_Ind, GMAT_Ind, Re_Admit_Term, Academic_Elig, Calc_Required_Term, Degree_Checkout, Admin_Hold, Re_Admit_Year, Calc_Required Year.

88. TRANSFERS:
SSAN*, Location_Code*, TRF_Hrs, TRF_Qual_Hrs, TRF_Qual_Pts, Begin_Date, Quarter, TRF_In_Date, TRF_End_Date, Old_Code.

LOGICAL DATABASE VIEW FOR:

DIRECTORATE OF ADMINISTRATION (AFIT/DA)

1. ACADEMIC TITLES:
Academic Title Code*, Academic Title.

2. AFIT CALENDAR:
Day*, Month*, Year*, Starting Time*, Activity, CC_Activity, Room*, Building*, POC, Remarks.

3. AFIT_COURSES:
Course*, Title, Description, Min_Credit, Max_Credit, Special_Grading, Section_Control, Remarks, Lec_Hrs, Lab_Hrs, Student_Limit, Special_Room_Type.

4. AFSC:
AFSC*, Description.

8. AWARDS:
SSAN*, Award, Date*.

10. BOOKS:
Title*, Author*, Publisher, No_Avail, Price, Office.

11. BOOK_ORDERS:
Author, Title, Order_No*, No_Ordered, Date, Status.

12. BOXES:
SSAN*, Box_No*.

16. COURSE_BOOKS:
Course*, Author*, Title*, No_Required, Quarter*, Year*.

17. COURSE_TIMES:
Course*, Starting_Time*, Day*, Quarter*, Year*, Room, Building, Course_Status, Wait_List_No, Begin_Date, End Date.

19. CURRENT_THESES:
SSAN*, Thesis_No, Advisors_SSAN, Reader1, Reader2.

20. DEGREES:
Degree_Code*, Description.

23. DUTY_OFFICER:
SSAN*, Duty, Credit_Date, Duty Date.

24. ED_CODES:

Ed Code*, Ed_Code_Title.

25. ED HIST:
SSAN*, College*, Begin Date, End Date, Degree_Code, Departure_Reason, Remarks.

28. EXTRA_DUTIES:
Extra_Duty_Code*, Extra_Duty_Title.

29. EXTRA_DUTY_ROSTER:
SSAN*, Extra_Duty_Code*, Date.

32. FAC_COURSES:
SSAN*, Course*, Quarter*, Year*, Instructor_No, Group_Contact_Hrs, Ind_Contact_Hrs.

34. FAC_REPLACEMENT:
Pos No*, Office, DAFSC, Rank, Name, Arrival_Date, Degree Required, Academic_Specialty, Status, DPhone.

36. FOREIGN STUDENTS DATA:
ID_No*, Service, Country, Funding.

41. LOCATIONS:
Location_Code*, Location_Name, Local_Address.

44. MAJCOMS:
MAJCOM*, MAJCOM_Title.

52. OFFICES:
Office*, Office Title.

55. PREREQUISITES:
Course*, Prerequisite*.

56. PRESENTATIONS:
SSAN*, Date*, Title, Location, Organization*, Subject.

60. PUBLICATIONS:
SSAN*, Title*, Date, Subject, Source*, Location, Pub_Kind, Co_Auth_Name.

63. RESIDENT_DATA:
SSAN*, Last_Organization, Duration, DOC, Emergency_Address.

64. ROOMS:
Room*, Building*, Max_Size, Special_Room_Type.

65. ROOM_SCHEDULE:
Day*, Month*, Year*, Starting_Time*, Ending_Time, Room, Building*, Function.

67. SECTION:
Section*, Graduation_Date, Entry_Date, No_Students, Leader_SSAN, Advisors_SSAN, Non_AF.

70. SPOUSES:
Sponsors_SSAN*, Name, DOB, No_Deps, Spouse_Type, Military_Spouse.

71. STAFF:
SSAN*, Name, Rank, HPhone, DPhone, Room, Office, Pos_No, Duty_Title, Date_Assigned, Departure_Date, Local_Address, DOB, DOR, Acad_Title_Code, DOAR, Marital_Status, Sex, PAFSC, DAFSC, Ethnic_Group, Service, Photo_Date, Security_Clearance, TMST, Previous_MAJCOM.

73. STAFF_POSITION_DATA:
Pos_No*, SSAN, Ed_Code_Required, Ed_Code_Assigned, Degree_Level_Required, Degree_Level_Assigned.

74. STUDENTS:
SSAN*, Name, Rank, PAFSC, Ed_Code, DOR, Sex, DOB, Advisors_SSAN, Entry_Date, DPhone, HPhone, Local Address, Foreign_Service_Ind, Service, Departure_Date, Security_Clearance, Duty_Location, Aero_Rating, Previous_MAJCOM, Marital_Status, TMST, Ethnic_Group, GRE, GMAT, Duty_Title, Section, Academic_Standing.

80. TDY:
Destination*, SSAN*, Departure_Date*, Return_Date, Estimated_Cost, Actual_Cost,Fees, Per_Diem, Travel.

81. THESES:
Thesis_No*, Title, Thesis_Sponsor, Classification, First_Author, Second_Author, Subject, Thesis_Pub_Date, DTIC_No, Cleared_For_Release.

82. THESIS_SPONSORS:
Thesis_Sponsor*, Location.

LOGICAL DATABASE VIEW FOR:

OFFICE OF PUBLIC AFFAIRS (AFIT/PA)

1. ACADEMIC_TITLES:
Academic_Title_Code*, Academic_Title.

2. AFIT_CALENDAR:
Day*, Month*, Year*, Starting_Time*, Activity, CC_Activity, Room*, Building*, POC, Remarks.

4. AFSC:
AFSC*, Description.

8. AWARDS:
SSAN*, Award, Date*.

19. CURRENT_THESES:
SSAN*, Thesis_No, Advisors_SSAN, Reader1, Reader2.

20. DEGREES:
Degree_Code*, Description.

21. DEGREE_SOUGHT:
SSAN*, Degree_Code, Major, Minor.

24. ED_CODES:
Ed_Code*, Ed_Code_Title.

25. ED_HIST:
SSAN*, College*, Begin_Date, End_Date, Degree_Code, Departure_Reason, Remarks.

36. FOREIGN_STUDENTS_DATA:
ID_No*, Service, Country, Funding.

41. LOCATIONS:
Location_Code*, Location_Name, Local_Address.

44. MAJCOMS:
MAJCOM*, MAJCOM_Title.

52. OFFICES:
Office*, Office_Title.

56. PRESENTATIONS:
SSAN*, Date*, Title, Location, Organization*, Subject.

57. PROFESSIONAL_ORGS:
Org_Abbrev*, Org_Name.

58. PROF_FAC_ORGS:
SSAN*, Org_Abbrev*.

59. PROMOTION_STATS:
Board_Date*, Board_Kind*, USAF, AFIT, USAF_Elig, AFIT_Elig.

60. PUBLICATIONS:
SSAN*, Title*, Date, Subject, Source*, Location, Pub_Kind, Co_Auth_Name.

61. RESIDENT_DATA:
SSAN*, Last_Organization, Duration, DOC, Emergency_Address.

67. SECTION:
Section*, Graduation_Date, Entry_Date, No_Students, Leader_SSAN, Advisors_SSAN, Non_AF.

70. SPOUSES:
Sponsors_SSAN*, Name, DOB, No_Deps, Spouse_Type, Military_Spouse.

71. STAFF:
SSAN*, Name, Rank, HPhone, DPhone, Room, Office, Pos No, Duty Title, Date Assigned, Departure Date, Local_Address, DOB, DOR, Acad_Title_Code, DOAR, Marital_Status, Sex, PAFSC, DAFSC, Ethnic_Group, Service, Photo_Date, Security_Clearance, TMST, Previous_MAJCOM.

74. STUDENTS:
SSAN*, Name, Rank, PAFSC, Ed_Code, DOR, Sex, DOB, Advisors_SSAN, Entry_Date, DPhone, HPhone, Local_Address, Foreign_Service_Ind, Service, Departure_Date, Security_Clearance, Duty_Location, Aero_Rating, Previous_MAJCOM, Marital_Status, TMST, Ethnic_Group, GRE, GMAT, Duty_Title, Section, Academic_Standing.

75. STUDENT_AWARDS:
SSAN*, Award, Date*.

76. STUDENT_HIST:
SSAN*, Name, Rank, PAFSC, Ed_Code, Sex, Entry_Date, Departure_Date, Service, Local_Address, Address_Date, Aero_Rating, Marital_Status, Ethnic_Group, GRE, GMAT, Degree_Code.

81. THESES:
Thesis_No*, Title, Thesis_Sponsor, Classification, First_Author, Second_Author, Subject, Thesis_Pub_Date, DTIC_No, Cleared_For_Release.

82. THESIS_SPONSORS:
    Thesis_Sponsor*, Location.

LOGICAL DATABASE VIEW FOR:

RESOURCE MANAGEMENT DIRECTORATE, BUDGET & ACCOUNTING (AFIT/ACB)

3. AFIT_COURSES:
Course*, Title, Description, Min_Credit, Max_Credit, Special_Grading, Section_Control, Remarks, Lec_Hrs, Lab_Hrs, Student_Limit, Special_Room_Type.

4. AFSC:
AFSC*, Description.

5. ALLOWANCE_PAYMENTS:
SSAN*, Allowance_Type*, Date_Paid*, Amount.

6. ALLOWANCE/PAYMENT_TYPES:
Allowance_Type*, Description.

11. BOOK_ORDERS:
Author, Title, Order_No*, No_Ordered, Date, Status.

14. CI_DATA:
SSAN*, Program, Output_AFSC, Degree_Code, Prior_GPA, Available_Date, Prior_AFIT, Departure_Reason, Suspense_Date, Legal_Residence, Thesis_Pay, Suspense_Action, Status, Begin_Date, C_GPA, Term_GPA.

15. CI_PROGRAMS:
Program*, Description.

17. COURSE_TIMES:
Course*, Starting_Time*, Day*, Quarter*, Year*, Room, Building, Course_Status, Wait_List_No, Begin_Date, End_Date.

18. CURRENT_CI_PROGRAMS:
Location_Code*, Program*, POC, DPhone.

25. ED_HIST:
SSAN*, College*, Begin_Date, End_Date, Degree_Code, Departure_Reason, Remarks.

27. EQUIPMENT:
Stock_No*, Description, Price, Date_Ordered*, Date_Received, Availability, Office.

36. FOREIGN_STUDENTS_DATA:
ID_No*, Service, Country, Funding.

37. GRADES:
SSAN*, Course*, Quarter*, Year*, Grade.

39. INSTITUTION_POCS:
Location_Code*, POC, DPhone.

40. INST_PAYMENTS:
Location_Code*, Date*, Payment_Type*, Amount.

41. LOCATIONS:
Location_Code*, Location_Name, Local_Address.

44. MAJCOMS:
MAJCOM*, MAJCOM_Title.

46. MED_PROGRAM:
SSAN*, MCAT_Score, MCAT_Score_Date, Degree_Code, Med_Program, Accession_Source.

48. MMEP:
Location_Code*, Supporting_Inst_Code.

50. MMEP_FUNDING:
Location_Code*, Fund_Type*, No_Students.

52. OFFICES:
Office*, Office_Title.

54. PCE:
Course*, Location_Code*, MAJCOM*, Student_Limit, No_Students.

66. SCHOOL/EWI_DATA:
Location_Code*, Servicing_AFO, Tuition_Rate.

74. STUDENTS:
SSAN*, Name, Rank, PAFSC, Ed_Code, DOR, Sex, DOB, Advisors_SSAN, Entry_Date, DPhone, HPhone, Local_Address, Foreign_Service_Ind, Service, Departure_Date, Security_Clearance, Duty_Location, Aero_Rating, Previous_MAJCOM, Marital_Status, TMST, Ethnic_Group, GRE, GMAT, Duty_Title, Section, Academic_Standing.

77. STUDENT_PAYMENTS:
SSAN*, ESA_No*, Date*, Payment_Type*, Amount.

80. TDY:
Destination*, SSAN*, Departure_Date*, Return_Date, Estimated_Cost, Actual_Cost,Fees, Per_Diem, Travel.

LOGICAL DATABASE VIEW FOR:

EDUCATION PLANS & OPERATIONS (AFIT/ED)

1. ACADEMIC_TITLES:
   Academic_Title_Code*, Academic_Title.

2. AFIT_CALENDAR:
   Day*, Month*, Year*, Starting_Time*, Activity, CC_Activity, Room*, Building*, POC, Remarks.

3. AFIT_COURSES:
   Course*, Title, Description, Min_Credit, Max_Credit, Special_Grading, Section_Control, Remarks, Lec_Hrs, Lab_Hrs, Student_Limit, Special_Room_Type.

4. AFSC:
   AFSC*, Description.

5. ALLOWANCE_PAYMENTS:
   SSAN*, Allowance_Type*, Date_Paid*, Amount.

6. ALLOWANCE/PAYMENT_TYPES:
   Allowance_Type*, Description.

7. AUDIO_VIS:
   Title*, Time, Producer*, Speaker, Subject, Production_Date.

8. AWARDS:
   SSAN*, Award, Date*.

9. BINDINGS:
   Title*, Volume, Month, Year, Color_No, Letter_Color.

10. BOOKS:
    Title*, Author*, Publisher, No_Avail, Price, Office.

11. BOOK_ORDERS:
    Author, Title, Order_No*, No_Ordered, Date, Status.

12. BOXES:
    SSAN*, Box_No*.

13. CHECKOUTS:
    Call_No*, Patron_Name, ID_No*, Due_Date.

14. CI_DATA:
    SSAN*, Program, Output_AFSC, Degree_Code, Prior_GPA, Available_Date, Prior_AFIT, Departure_Reason, Suspense_Date, Legal_Residence, Thesis_Pay,

Suspense_Action, Status, Begin_Date, C_GPA, Term_GPA.

15. CI_PROGRAMS:
Program*, Description.

16. COURSE_BOOKS:
Course*, Author*, Title*, No_Required, Quarter*, Year*.

17. COURSE_TIMES:
Course*, Starting_Time*, Day*, Quarter*, Year*, Room, Building, Course_Status, Wait_List_No, Begin_Date, End_Date.

18. CURRENT_CI_PROGRAMS:
Location_Code*, Program*, POC, DPhone.

19. CURRENT_THESES:
SSAN*, Thesis_No, Advisors_SSAN, Reader1, Reader2.

20. DEGREES:
Degree_Code*, Description.

21. DEGREE_SOUGHT:
SSAN*, Degree_Code, Major, Minor.

22. DE_DATA:
SSAN*, DAFSC, Funct_Acct_Code, DOS, Arrival_Date.

23. DUTY_OFFICER:
SSAN*, Duty, Credit_Date, Duty_Date.

24. ED_CODES:
Ed_Code*, Ed_Code_Title.

25. ED_HIST:
SSAN*, College*, Begin_Date, End_Date, Degree_Code, Departure_Reason, Remarks.

26. ED_PLANS:
SSAN*, Course*, Quarter*, Year*, Credits.

27. EQUIPMENT:
Stock_No*, Description, Price, Date_Ordered*, Date_Received, Availability, Office.

28. EXTRA_DUTIES:
Extra_Duty_Code*, Extra_Duty_Title.

29. EXTRA_DUTY_ROSTER:
SSAN*, Extra_Duty_Code*, Date.

30. FAC_BOARD:
SSAN*, Trouble_Term, Quarter*, Year*, Trouble_Year, Reason, Initial_Analysis, Board_Action, Success, Remarks.

31. FAC_BOARD_STUD_DATA:
SSAN*, Quarter*, Year*, UG_CGPA, GRE, GMAT, Term_GPA, C_GPA, A_Cred, B_Cred, C_Cred.

32. FAC_COURSES:
SSAN*, Course*, Quarter*, Year*, Instructor_No, Group_Contact_Hrs, Ind_Contact_Hrs.

33. FAC_ED:
SSAN*, Primary_Ed_Level, Secondary_Ed_Level, Special_Interest.

34. FAC_REPLACEMENT:
Pos_No*, Office, DAFSC, Rank, Name, Arrival_Date, Degree_Required, Academic_Specialty, Status, DPhone.

35. FAC_WORK:
SSAN*, Location*, Rank*, Begin_Date, End_Date.

36. FOREIGN_STUDENTS_DATA:
ID_No*, Service, Country, Funding.

37. GRADES:
SSAN*, Course*, Quarter*, Year*, Grade.

38. HOLDINGS:
Title*, Office*, Begin_Date, End_Date.

39. INSTITUTION_POCS:
Location_Code*, POC, DPhone.

40. INST_PAYMENTS:
Location_Code*, Date*, Payment_Type*, Amount.

41. LOCATIONS:
Location_Code*, Location_Name, Local_Address.

42. LOCKERS:
SSAN*, Locker_No*.

43. LIBRARY_BOOKS:
Call_No*, Title, Author.

44. MAJCOMS:
MAJCOM*, MAJCOM_Title.

45. MED_BOARD_CERTIFICATION:
SSAN*, Med_Spec_Code*, Certification_Date, ANAT_Score,

PHYS_Score, BIOCH_Score, PATH_Score, MICRO_Score, PHARM_Score, BEHSCI_Score, Score_Date.

46. MED_PROGRAM:
    SSAN*, MCAT_Score, MCAT_Score_Date, Degree_Code, Med_Program, Accession_Source.

47. MED_TOURS:
    SSAN*, Med_Spec_Code, Location_Code*, Begin_Date*, End_Date.

48. MMEP:
    Location_Code*, Supporting_Inst_Code.

49. MMEP_CLASSROOMS:
    Location_Code*, Room*, Building*.

50. MMEP_FUNDING:
    Location_Code*, Fund_Type*, No_Students.

51. NON_STUDENTS:
    Name, Card_No*, Organization, DPhone.

52. OFFICES:
    Office*, Office_Title.

53. PAST_COURSES:
    SSAN*, College*, Course*, Grade, Quarter*, Year*.

54. PCE:
    Course*, Location_Code*, MAJCOM*, Student_Limit, No_Students.

55. PREREQUISITES:
    Course*, Prerequisite*.

56. PRESENTATIONS:
    SSAN*, Date*, Title, Location, Organization*, Subject.

57. PROFESSIONAL_ORGS:
    Org_Abbrev*, Org_Name.

58. PROF_FAC_ORGS:
    SSAN*, Org_Abbrev*.

59. PROMOTION_STATS:
    Board_Date*, Board_Kind*, USAF, AFIT, USAF_Elig, AFIT_Elig.

60. PUBLICATIONS:
    SSAN*, Title*, Date, Subject, Source*, Location, Pub_Kind, Co_Auth_Name.

61. QUOTAS:
Ed_Code*, Quota_Type*, YR1, YR2, YR3, YR4, YR5.

62. RESERVES:
Call_No*, FAC_SSAN*, Date_Reserved*.

63. RESIDENT_DATA:
SSAN*, Last_Organization, Duration, DOC, Emergency_Address.

64. ROOMS:
Room*, Building*, Max_Size, Special_Room_Type.

65. ROOM_SCHEDULE:
Day*, Month*, Year*, Starting_Time*, Ending_Time, Room, Building*, Function.

66. SCHOOL/EWI_DATA:
Location_Code*, Servicing_AFO, Tuition_Rate.

67. SECTION:
Section*, Graduation_Date, Entry_Date, No_Students, Leader_SSAN, Advisors_SSAN, Non_AF.

68. SHORT_COURSE:
Course*, Title, Begin_Date*, End_Date, Room*, Building*, Starting_Time, Description.

69. SHORT_STUDENTS:
SSAN*, Course*, Begin_Date*, Name, Rank, Organization, DPhone

70. SPOUSES:
Sponsors_SSAN*, Name, DOB, No_Deps, Spouse_Type, Military_Spouse.

71. STAFF:
SSAN*, Name, Rank, HPhone, DPhone, Room, Office, Pos_No, Duty_Title, Date_Assigned, Departure_Date, Local_Address, DOB, DOR, Acad_Title_Code, DOAR, Marital_Status, Sex, PAFSC, DAFSC, Ethnic_Group, Service, Photo_Date, Security_Clearance, TMST, Previous_MAJCOM.

72. STAFF_DE_DATA:
SSAN*, Course_Directed*.

73. STAFF_POSITION_DATA:
Pos_No*, SSAN, Ed_Code_Required, Ed_Code_Assigned, Degree_Level_Required, Degree_Level_Assigned.

74. STUDENTS:
SSAN*, Name, Rank, PAFSC, Ed_Code, DOR, Sex, DOB,

Advisors_SSAN, Entry_Date, DPhone, HPhone, Local_Address, Foreign_Service_Ind, Service, Departure_Date, Security_Clearance, Duty_Location, Aero_Rating, Previous_MAJCOM, Marital_Status, TMST, Ethnic_Group, GRE, GMAT, Duty_Title, Section, Academic_Standing.

75. STUDENT_AWARDS:
SSAN*, Award, Date*.

76. STUDENT_HIST:
SSAN*, Name, Rank, PAFSC, Ed_Code, Sex, Entry_Date, Departure_Date, Service, Local_Address, Address_Date, Aero_Rating, Marital_Status, Ethnic_Group, GRE, GMAT, Degree_Code.

77. STUDENT_PAYMENTS:
SSAN*, ESA_No*, Date*, Payment_Type*, Amount.

78. STUDENT_SCHED:
SSAN*, Course*.

79. SUBSCRIPTIONS:
Title*, Check_In, Renewal_Date, Price, Expiration_Date, Requester, Begin_Date, Follow_Up_Date, Follow_Up_Remarks.

80. TDY:
Destination*, SSAN*, Departure_Date*, Return_Date, Estimated_Cost, Actual_Cost,Fees, Per_Diem, Travel.

81. THESES:
Thesis_No*, Title, Thesis_Sponsor, Classification, First_Author, Second_Author, Subject, Thesis_Pub_Date, DTIC_No, Cleared_For_Release.

82. THESIS_SPONSORS:
Thesis_Sponsor*, Location.

RELATIONS PERTAINING TO THE REGISTRARS INFORMATION

83. COURSE_GRADES:
SSAN*, Course*, Grade_Type, Grade, Prior_Grade, Registered_Hrs, Earned_Hrs, Qual_Pts, Drop_Reason, Date.

84. COURSE_STATS:
Course*, Proj_Enrollment, Min_Enrollment, No_Enrolled, Demand, Drops, Wait_List_No, No_Not_Registered,

No_Added, No_Audits.

85. GMAT_SCORES:
SSAN*, Total.

86. GRE_SCORES:
SSAN*, Verbal, Quantitative, Total.

87. RR_DATA:
SSAN*, Manning_Code, Output_MAJCOM, UG_CGPA, Degree_Code, Prior_AFIT, Folder_Location, Departure_Reason, Admission_Action, Transfer_Ind, Prev_CPA, C_GPA, GRE_Ind, GMAT_Ind, Re_Admit_Term, Academic_Elig, Calc_Required_Term, Degree_Checkout, Admin_Hold, Re_Admit_Year, Calc_Required_Year.

88. TRANSFERS:
SSAN*, Location_Code*, TRF_Hrs, TRF_Qual_Hrs, TRF_Qual_Pts, Begin_Date, Quarter, TRF_In_Date, TRF_End_Date, Old_Code.

LOGICAL DATABASE VIEW FOR:

SCHOOL OF SYSTEMS & LOGISTICS (AFIT/LS)

1. ACADEMIC_TITLES:
Academic_Title_Code*, Academic_Title.

2. AFIT_CALENDAR:
Day*, Month*, Year*, Starting_Time*, Activity, CC_Activity, Room*, Building*, POC, Remarks.

3. AFIT_COURSES:
Course*, Title, Description, Min_Credit, Max_Credit, Special_Grading, Section_Control, Remarks, Lec_Hrs, Lab_Hrs, Student_Limit, Special_Room_Type.

4. AFSC:
AFSC*, Description.

5. ALLOWANCE_PAYMENTS:
SSAN*, Allowance_Type*, Date_Paid*, Amount.

6. ALLOWANCE/PAYMENT_TYPES:
Allowance_Type*, Description.

8. AWARDS:
SSAN*, Award, Date*.

10. BOOKS:
Title*, Author*, Publisher, No_Avail, Price, Office.

11. BOOK_ORDERS:
Author, Title, Order_No*, No_Ordered, Date, Status.

12. BOXES:
SSAN*, Box_No*.

16. COURSE_BOOKS:
Course*, Author*, Title*, No_Required, Quarter*, Year*.

17. COURSE_TIMES:
Course*, Starting_Time*, Day*, Quarter*, Year*, Room, Building, Course_Status, Wait_List_No, Begin_Date, End_Date.

19. CURRENT_THESES:
SSAN*, Thesis_No, Advisors_SSAN, Reader1, Reader2.

21. DEGREE_SOUGHT:
SSAN*, Degree_Code, Major, Minor.

24. ED_CODES:
Ed_Code*, Ed_Code_Title.

25. ED_HIST:
SSAN*, College*, Begin_Date, End_Date, Degree_Code, Departure_Reason, Remarks.

26. ED_PLANS:
SSAN*, Course*, Quarter*, Year*, Credits.

27. EQUIPMENT:
Stock_No*, Description, Price, Date_Ordered*, Date_Received, Availability, Office.

28. EXTRA_DUTIES:
Extra_Duty_Code*, Extra_Duty_Title.

29. EXTRA_DUTY_ROSTER:
SSAN*, Extra_Duty_Code*, Date.

30. FAC_BOARD:
SSAN*, Trouble_Term, Quarter*, Year*, Trouble_Year, Reason, Initial_Analysis, Board_Action, Success, Remarks.

31. FAC_BOARD_STUD_DATA:
SSAN*, Quarter*, Year*, UG_CGPA, GRE, GMAT, Term_GPA, C_GPA, A_Cred, B_Cred, C_Cred.

32. FAC_COURSES:
SSAN*, Course*, Quarter*, Year*, Instructor_No, Group_Contact_Hrs, Ind_Contact_Hrs.

33. FAC_ED:
SSAN*, Primary_Ed_Level, Secondary_Ed_Level, Special_Interest.

34. FAC_REPLACEMENT:
Pos_No*, Office, DAFSC, Rank, Name, Arrival_Date, Degree_Required, Academic_Specialty, Status, DPhone.

35. FAC_WORK:
SSAN*, Location*, Rank*, Begin_Date, End_Date.

36. FOREIGN_STUDENTS_DATA:
ID_No*, Service, Country, Funding.

37. GRADES:
SSAN*, Course*, Quarter*, Year*, Grade.

41. LOCATIONS:
Location_Code*, Location_Name, Local_Address.

42. LOCKERS:
SSAN*, Locker_No*.

44. MAJCOMS:
MAJCOM*, MAJCOM_Title.

52. OFFICES:
Office*, Office_Title.

53. PAST_COURSES:
SSAN*, College*, Course*, Grade, Quarter*, Year*.

54. PCE:
Course*, Location_Code*, MAJCOM*, Student_Limit, No_Students.

55. PREREQUISITES:
Course*, Prerequisite*.

56. PRESENTATIONS:
SSAN*, Date*, Title, Location, Organization*, Subject.

57. PROFESSIONAL_ORGS:
Org_Abbrev*, Org_Name.

58. PROF_FAC_ORGS:
SSAN*, Org_Abbrev*.

60. PUBLICATIONS:
SSAN*, Title*, Date, Subject, Source*, Location, Pub_Kind, Co_Auth_Name.

63. RESIDENT_DATA:
SSAN*, Last_Organization, Duration, DOC, Emergency_Address.

64. ROOMS:
Room*, Building*, Max_Size, Special_Room_Type.

65. ROOM_SCHEDULE:
Day*, Month*, Year*, Starting_Time*, Ending_Time, Room, Building*, Function.

67. SECTION:
Section*, Graduation_Date, Entry_Date, No_Students, Leader_SSAN, Advisors_SSAN, Non_AF.

68. SHORT_COURSE:
Course*, Title, Begin_Date*, End_Date, Room*, Building*, Starting_Time, Description.

69. SHORT_STUDENTS:
SSAN*, Course*, Begin_Date*, Name, Rank, Organization, DPhone

70. SPOUSES:
Sponsors_SSAN*, Name, DOB, No_Deps, Spouse_Type, Military_Spouse.

71. STAFF:
SSAN*, Name, Rank, HPhone, DPhone, Room, Office, Pos_No, Duty_Title, Date_Assigned, Departure_Date, Local_Address, DOB, DOR, Acad_Title_Code, DOAR, Marital_Status, Sex, PAFSC, DAFSC, Ethnic_Group, Service, Photo_Date, Security_Clearance, TMST, Previous_MAJCOM.

73. STAFF_POSITION_DATA:
Pos_No*, SSAN, Ed_Code_Required, Ed_Code_Assigned, Degree_Level_Required, Degree_Level_Assigned.

74. STUDENTS:
SSAN*, Name, Rank, PAFSC, Ed_Code, DOR, Sex, DOB, Advisors_SSAN, Entry_Date, DPhone, HPhone, Local_Address, Foreign_Service_Ind, Service, Departure_Date, Security_Clearance, Duty_Location, Aero_Rating, Previous_MAJCOM, Marital_Status, TMST, Ethnic_Group, GRE, GMAT, Duty_Title, Section, Academic_Standing.

75. STUDENT_AWARDS:
SSAN*, Award, Date*.

76. STUDENT_HIST:
SSAN*, Name, Rank, PAFSC, Ed_Code, Sex, Entry_Date, Departure_Date, Service, Local_Address, Address_Date, Aero_Rating, Marital_Status, Ethnic_Group, GRE, GMAT, Degree_Code.

78. STUDENT_SCHED:
SSAN*, Course*.

80. TDY:
Destination*, SSAN*, Departure_Date*, Return_Date, Estimated_Cost, Actual_Cost,Fees, Per_Diem, Travel.

81. THESES:
Thesis_No*, Title, Thesis_Sponsor, Classification, First_Author, Second_Author, Subject, Thesis_Pub_Date, DTIC_No, Cleared_For_Release.

82. THESIS_SPONSORS:
Thesis_Sponsor*, Location.

RELATIONS PERTAINING TO THE REGISTRARS INFORMATION

83. COURSE_GRADES:
    SSAN*, Course*, Crade_Type, Grade, Prior_Grade, Registered_Hrs, Earned_Hrs, Qual_Pts, Drop_Reason, Date.

84. COURSE_STATS:
    Course*, Proj_Enrollment, Min_Enrollment, No_Enrolled, Demand, Drops, Wait_List_No, No_Not_Registered, No_Added, No_Audits.

85. GMAT_SCORES:
    SSAN*, Total.

86. GRE_SCORES:
    SSAN*, Verbal, Quantitative, Total.

LOGICAL DATABASE VIEW FOR:

SCHOOL OF ENGINEERING (AFIT/EN)

1. ACADEMIC_TITLES:
Academic_Title_Code*, Academic_Title.

2. AFIT_CALENDAR:
Day*, Month*, Year*, Starting_Time*, Activity, CC_Activity, Room*, Building*, POC, Remarks.

3. AFIT_COURSES:
Course*, Title, Description, Min_Credit, Max_Credit, Special_Grading, Section_Control, Remarks, Lec_Hrs, Lab_Hrs, Student_Limit, Special_Room_Type.

4. AFSC:
AFSC*, Description.

5. ALLOWANCE_PAYMENTS:
SSAN*. Allowance_Type*, Date_Paid*, Amount.

6. ALLOWANCE/PAYMENT_TYPES:
Allowance_Type*, Description.

7. AUDIO_VIS:
Title*, Time, Producer*, Speaker, Subject, Production_Date.

8. AWARDS:
SSAN*, Award, Date*.

10. BOOKS:
Title*, Author*, Publisher, No_Avail, Price, Office.

11. BOOK_ORDERS:
Author, Title, Order_No*, No_Ordered, Date, Sta'us.

12. BOXES:
SSAN*, Box_No*.

16. COURSE_BOOKS:
Course*, Author*, Title*, No_Required, Quarter*, Year*.

17. COURSE_TIMES:
Course*, Starting_Time*, Day*, Quarter*, Year*, Room, Building, Course_Status, Wait_List_No, Begin_Date, End_Date.

19. CURRENT_THESES:

SSAN*, Thesis_No, Advisors_SSAN, Reader1, Reader2.

20. DEGREES:
Degree_Code*, Description.

21. DEGREE_SOUGHT:
SSAN*, Degree_Code, Major, Minor.

24. ED_CODES:
Ed_Code*, Ed_Code_Title.

25. ED_HIST:
SSAN*, College*, Begin_Date, End_Date, Degree_Code, Departure_Reason, Remarks.

26. ED_PLANS:
SSAN*, Course*, Quarter*, Year*, Credits.

27. EQUIPMENT:
Stock_No*, Description, Price, Date_Ordered*, Date_Received, Availability, Office.

28. EXTRA_DUTIES:
Extra_Duty_Code*, Extra_Duty_Title.

29. EXTRA_DUTY_ROSTER:
SSAN*, Extra_Duty_Code*, Date.

30. FAC_BOARD:
SSAN*, Trouble_Term, Quarter*, Year*, Trouble_Year, Reason, Initial_Analysis, Board_Action, Success, Remarks.

31. FAC_BOARD_STUD_DATA:
SSAN*, Quarter*, Year*, UG_CGPA, GRE, GMAT, Term_GPA, C_GPA, A_Cred, B_Cred, C_Cred.

32. FAC_COURSES:
SSAN*, Course*, Quarter*, Year*, Instructor_No, Group_Contact_Hrs, Ind_Contact_Hrs.

33. FAC_ED:
SSAN*, Primary_Ed_Level, Secondary_Ed_Level, Special_Interest.

34. FAC_REPLACEMENT:
Pos_No*, Office, DAFSC, Rank, Name, Arrival_Date, Degree_Required, Academic_Specialty, Status, DPhone.

35. FAC_WORK:
SSAN*, Location*, Rank*, Begin_Date, End_Date.

36. FOREIGN_STUDENTS_DATA:
ID_No*, Service, Country, Funding.

37. GRADES:
SSAN*, Course*, Quarter*, Year*, Grade.

41. LOCATIONS:
Location_Code*, Location_Name, Local_Address.

42. LOCKERS:
SSAN*, Locker_No*.

44. MAJCOMS:
MAJCOM*, MAJCOM_Title.

52. OFFICES:
Office*, Office_Title.

53. PAST_COURSES:
SSAN*, College*, Course*, Grade, Quarter*. Year*.

54. PCE:
Course*, Location_Code*, MAJCOM*, Student_Limit, No_Students.

55. PREREQUISITES:
Course*, Prerequisite*.

56. PRESENTATIONS:
SSAN*, Date*, Title, Location, Organization*, Subject.

57. PROFESSIONAL_ORGS:
Org_Abbrev*, Org_Name.

58. PROF_FAC_ORGS:
SSAN*, Org_Abbrev*.

60. PUBLICATIONS:
SSAN*, Title*, Date, Subject, Source*, Location, Pub_Kind, Co_Auth_Name.

63. RESIDENT_DATA:
SSAN*, Last_Organization, Duration, DOC, Emergency_Address.

64. ROOMS:
Room*, Building*, Max_Size, Special_Room_Type.

65. ROOM_SCHEDULE:
Day*, Month*, Year*, Starting_Time*, Ending_Time, Room, Building*, Function.

67. SECTION:

Section*, Graduation_Date, Entry_Date, No_Students, Leader_SSAN, Advisors_SSAN, Non_AF.

68. SHORT_COURSE:
Course*, Title, Begin_Date*, End_Date, Room*, Building*, Starting_Time, Description.

69. SHORT_STUDENTS:
SSAN*, Course*, Begin_Date*, Name, Rank, Organization, DPhone

70. SPOUSES:
Sponsors_SSAN*, Name, DOB, No_Deps, Spouse_Type, Military_Spouse.

71. STAFF:
SSAN*, Name, Rank, HPhone, DPhone, Room, Office, Pos_No, Duty_Title, Date_Assigned, Departure_Date, Local_Address, DOB, DOR, Acad_Title_Code, DOAR, Marital_Status, Sex, PAFSC, DAFSC, Ethnic_Group, Service, Photo_Date, Security_Clearance, TMST, Previous_MAJCOM.

73. STAFF_POSITION_DATA:
Pos_No*, SSAN, Ed_Code_Required, Ed_Code_Assigned, Degree_Level_Required, Degree_Level_Assigned.

74. STUDENTS:
SSAN*, Name, Rank, PAFSC, Ed_Code, DOR, Sex, DOB, Advisors_SSAN, Entry_Date, DPhone, HPhone, Local_Address, Foreign_Service_Ind, Service, Departure_Date, Security_Clearance, Duty_Location, Aero_Rating, Previous_MAJCOM, Marital_Status, TMST, Ethnic_Group, GRE, GMAT, Duty_Title, Section, Academic_Standing.

75. STUDENT_AWARDS:
SSAN*, Award, Date*.

76. STUDENT_HIST:
SSAN*, Name, Rank, PAFSC, Ed_Code, Sex, Entry_Date, Departure_Date, Service, Local_Address, Address_Date, Aero_Rating, Marital_Status, Ethnic_Group, GRE, GMAT, Degree_Code.

78. STUDENT_SCHED:
SSAN*, Course*.

80. TDY:
Destination*, SSAN*, Departure_Date*, Return_Date, Estimated_Cost, Actual_Cost,Fees, Per_Diem, Travel.

81. THESES:

Thesis_No*, Title, Thesis_Sponsor, Classification, First_Author, Second_Author, Subject, Thesis_Pub_Date, DTIC_No, Cleared_For_Release.

82. THESIS_SPONSORS:
Thesis_Sponsor*, Location.

RELATIONS PERTAINING TO THE REGISTRARS INFORMATION

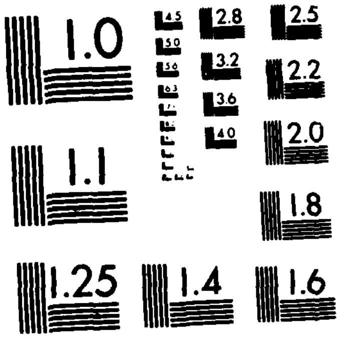
83. COURSE_GRADES:
SSAN*, Course*, Grade_Type, Grade, Prior_Grade, Registered_Hrs, Earned_Hrs, Qual_Pts, Drop_Reason, Date.

84. COURSE_STATS:
Course*, Proj_Enrollment, Min_Enrollment, No_Enrolled, Demand, Drops, Wait_List_No, No_Not_Registered, No_Added, No_Audits.

85. GMAT_SCORES:
SSAN*, Total.

86. GRE_SCORES:
SSAN*, Verbal, Quantitative, Total.

AD-A124 647 ANALYSIS OF INFORMATION REQUIREMENTS AND DESIGN OF THE 3/4
CONSOLIDATED AFIT D..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..
UNCLASSIFIED J S RICKS ET AL. DEC 82 AFIT/GCS/EE/82D-11 F/G 5/2 NL


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

LOGICAL DATABASE VIEW FOR:

AFIT BOOKSTORE

2. AFIT CALENDAR:
Day*, Month*, Year*, Starting Time*, Activity, CC Activity, Room*, Building*, POC, Remarks.

10. BOOKS:
Title*, Author*, Publisher, No Avail, Price, Office.

11. BOOK ORDERS:
Author, Title, Order No*, No Ordered, Date, Status.

16. COURSE BOOKS:
Course*, Author*, Title*, No_Required, Quarter*, Year*.

27. EQUIPMENT:
Stock No*, Description, Price, Date Ordered*, Date Received, Availability, Office.

32. FAC COURSES:
SSAN*, Course*, Quarter*, Year*, Instructor No, Group Contact Hrs, Ind Contact Hrs.

LOGICAL DATABASE VIEW FOR:

PROFESSIONAL DEVELOPMENT (AFIT/NR)

1. ACADEMIC_TITLES:
Academic Title Code*, Academic_Title.

2. AFIT CALENDAR:
Day*, Month*, Year*, Starting Time*, Activity, CC Activity, Room*, Building*, POC, Remarks.

4. AFSC:
AFSC*, Description.

8. AWARDS:
SSAN*, Award, Date*.

19. CURRENT THESES:
SSAN*, Thesis_No, Advisors_SSAN, Reader1, Reader2.

27. EQUIPMENT:
Stock_No*, Description, Price, Date_Ordered*, Date_Received, Availability, Office.

32. FAC_COURSES:
SSAN*, Course*, Quarter*, Year*, Instructor_No, Group_Contact_Hrs, Ind_Contact_Hrs.

33. FAC_ED:
SSAN*, Primary_Ed_Level, Secondary_Ed_Level, Special_Interest.

34. FAC_REPLACEMENT:
Pos_No*, Office, DAFSC, Rank, Name, Arrival_Date, Degree_Required, Academic_Specialty, Status, DPhone.

35. FAC_WORK:
SSAN*, Location*, Rank*, Begin_Date, End_Date.

44. MAJCOMS:
MAJCOM*, MAJCOM_Title.

52. OFFICES:
Office*, Office_Title.

56. PRESENTATIONS:
SSAN*, Date*, Title, Location, Organization*, Subject.

57. PROFESSIONAL_ORGS:

Org_Abbrev*, Org_Name.

58. PROF_FAC_ORGS:
SSAN*, Org_Abbrev*.

59. PROMOTION_STATS:
Board_Date*, Board_Kind*, USAF, AFIT, USAF_Elig, AFIT_Elig.

60. PUBLICATIONS:
SSAN*, Title*, Date, Subject, Source*, Location, Pub_Kind, Co_Auth_Name.

64. ROOMS:
Room*, Building*, Max_Size, Special_Room_Type.

65. ROOM_SCHEDULE:
Day*, Month*, Year*, Starting_Time*, Ending_Time, Room, Building*, Function.

70. SPOUSES:
Sponsors_SSAN*, Name, DOB, No_Deps, Spouse_Type, Military_Spouse.

71. STAFF:
SSAN*, Name, Rank, HPhone, DPhone, Room, Office, Pos_No, Duty_Title, Date_Assigned, Departure_Date, Local_Address, DOB, DOR, Acad_Title_Code, DOAR, Marital_Status, Sex, PAFSC, DAFSC, Ethnic_Group, Service, Photo_Date, Security_Clearance, TMST, Previous_MAJCOM.

73. STAFF_POSITION_DATA:
Pos_No*, SSAN, Ed_Code_Required, Ed_Code_Assigned, Degree_Level_Required, Degree_Level_Assigned.

80. TDY:
Destination*, SSAN*, Departure_Date*, Return_Date, Estimated_Cost, Actual_Cost,Fees, Per_Diem, Travel.

81. THESES:
Thesis_No*, Title, Thesis_Sponsor, Classification, First_Author, Second_Author, Subject, Thesis_Pub_Date, DTIC_No, Cleared_For_Release.

82. THESIS_SPONSORS:
Thesis_Sponsor*, Location.

RELATION	QTY	ESTIMATION FACTORS	QTY SOURCE
1	12	CONSTANT	SE
2	2400	10/day * 5 days/wk * 12 wks/qtr * 4 qtr/yr (see note 1)	AE
3	563	Total resident courses in the AFIT catalog; CONSTANT	AC
4	549	MPC AFSCs; CONSTANT	AC
5	931	AFIT/EN only; CONSTANT	SE
6	6	3/EN + 3/CI; CONSTANT	SE
7	250	CONSTANT	SE
8	250	CONSTANT	AE
9	2000	CONSTANT	SE
10	20000	200 courses/qtr * 25 books/course * 4 qtr/yr	AE
11	800	1/course * 200 courses/qtr * 4 qtr/yr	AE
12	1013	EN & LS only; CONSTANT	AC
13	500	Daily average from LD; CONSTANT	SE
14	4058	All CI students listed on the 8 Sep 82 SAR; CONSTANT	SE
15	2400	400 institutions * 6 programs/institution CONSTANT	SE
16	1600	2 books/course * 200 courses/qtr * 4 qtr/yr	AE
17	1600	2 times/course * 200 courses/qtr * 4qtr/yr	AE
18	121	CI catalog; CONSTANT	AC
19	500	CONSTANT	AE
20	500	CONSTANT	AE
21	5200	1063 EN & LS students + 4058 CI students + ROF; Sep 82 SAR; CONSTANT	SE
22	2140	2096 DE students + 40 DE staff + ROF; CONSTANT	SE
23	134	CONSTANT	SE
24	100	CONSTANT	SE
25	1000	995 EN & LS resident students + ROF; CONSTANT; (Note: Part time students not counted); 8 Sep 82 SAR	SE
26	20000	1000 EN & LS students * 20 entries/student; CONSTANT	AE
27	500	CONSTANT	AE
28	255	DA duty roster; CONSTANT	AC
29	270	DA duty roster; CONSTANT	AC
30	100	CONSTANT	SE
31	250	100/year * 2.5/student; CONSTANT	SE
32	2700	225 EN, LS & DE faculty (includes ROF) * 3 courses per faculty/qtr * 4 qtr/yr	SE
33	225	225 Total faculty (includes ROF); CONSTANT	SE
34	25	CONSTANT	AE
35	1800	225 faculty members * 8/faculty member CONSTANT	SE
36	34	8 Sep 82 SAR; CONSTANT	SE
37	16000	1000 resident students * 4 courses/student * 4 qtrs/yr	AE
38	1000	CONSTANT	SE
39	400	1/institute * 400 institutes; CONSTANT	SE

40	2400	400 institutes * 6/institute; CONSTANT	SE
41	40000	CONSTANT	AE
42	795	CONSTANT	AC
43	81320	80000 books with call no.s + (110 new books/mo. * 12 mo./yr.)	SE
44	14	CONSTANT	AE
45	1000	(440 CIM students+ROF) *2 Med Certifications/student; 8 Sep 82 SAR; CONSTANT	SE
46	1867	440 CIM students + 1427 AFHPSP students; 8 Sep 82 SAR; CONSTANT	SE
47	5600	(440 CIM students + 1427 HPSP students) * 3 tours/student; CONSTANT	AE
48	6	CONSTANT	SE
49	10	CONSTANT	AE
50	24	6 MMEP programs * 4 fund types/program	Z
51	700	CONSTANT	SE
52	100	AFIT Staff Directory, Oct 81; CONSTANT	AC
53	----	NOT INCLUDED IN ESTIMATION	AE
54	128	AFIT catalog (LS + DE + EN + PCE courses); CONSTANT	AC
55	1029	((563 resid. courses * 80% with prerequisites) * 2 prerequisites/course) + ((128 PCE courses * 50 % with prerequisites) * 2 prerequisites/course) + ROF	AE
56	2700	225 faculty members * 3/member/qtr * 4 qtr/yr	AE
57	200	CONSTANT	SE
58	1125	225 faculty members * 5/person; CONSTANT	SE
59	100	CONSTANT	AE
60	675	225 faculty members * 3/person; CONSTANT	AE
61	200	(100 quotas (1/ED Code)) * 2 types of quotas; CONSTANT	SE
62	3000	CONSTANT	SE
63	1000	995 resident students + ROF; CONSTANT	SE
64	130	LS + EN + Bldg 125 schedulable rooms + ROF; CONSTANT	AC
65	135200	130 rooms * 4 events/room/day * 260 school days/yr; CONSTANT	AE
66	475	400 schools + 75 EWI locations; CONSTANT	SE
67	52	CONSTANT	SE
68	128	AFIT catalog; EN + LS + DE PCE courses; CONSTANT	AC
69	13000	2096 DE ('82) + 9000 LS ('83) + 1800 EN ('82) + ROF; CONSTANT	SE
70	1000	CONSTANT	AE
71	578	All staff listed on 8 Sep 82 SAR; CONSTANT	SE
72	30	30 courses * 1 director/course; CONSTANT	SE
73	578	Same as 71	SE
74	5058	995 resident students + ROF + 4058 CI students; 8 Sep 82 SAR; CONSTANT	SE
75	250	CONSTANT	AE
76	----	NOT INCLUDED IN ESTIMATE	AE
77	10000	CONSTANT	SE
78	16000	1000 students * 4 courses/student/qtr *	AE

		4 qtrs/yr; CONSTANT	
79	1400	CONSTANT	SE
80	1125	225 faculty members * 5 TDYs/member/yr; CONSTANT	AE
81	5400	5121 + 1 yr estimated growth + ROF	SE
82	50	CONSTANT	AE
83		NOT INCLUDED IN ESTIMATE	AE
84		NOT INCLUDED IN ESTIMATE	AE
85		NOT INCLUDED IN ESTIMATE	AE
86		NOT INCLUDED IN ESTIMATE	AE
87		NOT INCLUDED IN ESTIMATE	AE
88		NOT INCLUDED IN ESTIMATE	AE

NOTES

1. * indicates multiplication, e.g. 4 * 5 means 4 times 5. 2. These estimates are for the database at the end of the first year of operation.

COMMENTS ON THE TERMS USED IN THIS LIST MAY BE FOUND ON THE FOLLOWING PAGE

TERMINOLOGY EXPLANATION

STAFF ESTIMATE (SE)
Some staff or faculty member from the appropriate department provided either the estimate shown or the major input for arriving at it.

AUTHORS ESTIMATE (AE)
The authors estimated the figure(s) based upon available information or experience.

ACTUAL COUNT (AC)
The estimate shown was arrived at by counting the appropriate raw data.

8 Sep 82 SAR
The 8 Sep 1982 copy of the Significant Activities Report published by AFIT/DP for AFIT/DA.

CONSTANT
The estimate represents a value which is relatively independent of time. It could be an average or an actual value which remains steady.

ROUND OFF FACTOR (ROF)
Any estimate which has this factor was rounded up to a value which was easier to use in calulations. (e.g. 900, 990 and 995 all would be rounded up to 1000)

Appendix I,I

FUNCTIONAL DEPENDENCIES

RELATION	FUNC'IONAL DEPENDENCIES	IMPORTANT ASSUMPTIONS
1	1->2,2->1 (see notes 1 & 2)	
2	123478->569(10) (see note 3)	A given ACTIVITY is not always a CC_ACTIVITY A given ACTIVITY does not always have the same POC A given ACTIVITY does not always have the same REMARKS

3	1->23456789(10)(11)(12)
	3->12456789(10)(11)(12)
		TITLE is not necessarily unigue

4	1->2, 2->1	
5	123->4	AMOUNT may vary for a given ALLOWANCE_TYPE
6	1->2, 2->1	
7	13->2456	TITLE is not necessarily unique
8	13->2	
9	1->23456	
10	12->3456	TITLE is not necessarily unique
11	3->12456, 12->3456	TITLE is not necessarily unique
12	12->12	
13	13->2	PATRON_NAME not necessarily unique
14	1->23456789(10)(11)(12)(13)(14)(15)(16)	PROGRAM does not uniquely determine OUTPUT_AFSC OUTPUT_AFSC does not uniquely determine PROGRAM PROGRAM does not uniquely determine DEGREE_CODE
15	1->2, 2->1	
16	12354->4	
17	12345->6789(10)(11)	
18	12->34	POC does not uniquely determine DPHONE DPHONE does not uniquely determine POC
19	1->2345, 2->1345	THESIS_NO is unique
20	1->2, 2->1	
21	1->234	At AFIT every student seeks a max of 1 degree at any given time
22	1->2345	
23	1->234	
24	1->2, 2->1	ED_CODE does not uniquely determine SCHOOL SCHOOL does not uniquely determine ED_CODE
25	12->34567	
26	1234->5	COURSE, QUARTER, & YEAR do not uniquely determine CREDITS
27	14->23567	STOCK_NO does not uniquely determine DESCRIPTION, PRICE, and AVAILABILITY DESCRIPTION and PRICE do not uniquely determine STOCK_NO
28	1->2, 2->1	
29	12->3	
30	134->256789(10)	More than one FAC_BOARD tuple may exist for a given student
31	123->456789(10)(11)	
32	1234->567	
33	1->234	
34	1->23456789(10)	NAME not necessarily unique
35	123->45	
36	1->234	
37	1234->5	A given student may repeat a COURSE

```
38          12->34                                          TITLE not necessarily unique
39          1->23                                           POC does not uniquely determine DPHONE
                                                            DPHONE  does not uniquely determine POC
40          123->4                                          AMOUNT may vary for a given PAYMENT_TYPE
41          1->23, 23->1                                    LOCATION_NAME does not necessarily determine LOCAL_ADDRESS
                                                            LOCAL_ADDRESS does not necessarily determine LOCATION_NAME
42          12->12                                          SSAN does not uniquely determine LOCKER_NO because it is
                                                            assumed empty lockers are stored in tuples as having
                                                            all zeros in the SSAN attribute
43          1->23                                           TITLE not necessarily unique
44          1->2, 2->1
45          12->3456789(10)(11)
46          1->23456
47          134->25                                         More than one MED_TOUR possibe at a given LOCATION
48          1->2                                            A given MMEP program has only one supporting institution
                                                            A given institution may support more than one MMEP program
49          123->123
50          12->3
51          2-> 134                                         NAME not necessarily unique
                                                            ORGANIZATION does not uniquely determine DPHONE
                                                            DPHONE does not necessarily uniquely determine ORGANIZATION
52          1->2, 2->1
53          12356->4
54          123->45
55          12->12
56          125->346                                        TITLE not necessarily unique
                                                            2 PRESENTATIONS in different places on same day possible
57          1->2, 2->1
58          12->12
59          12->3456
60          125->34678                                      Possibly some articles published in more than one SOURCE
                                                            No repeat publications in same SOURCE
61          12->34567
62          12->3, 23->1                                    Only latest faculty reservation stored
63          1->2345
64          12->34
65          12347->568
66          1->23
67          1->234567
68          1356->2478, 8356->1247                          BEGIN_DATE needed in key to distinguish between the
                                                            records of the same course offered on different dates
                                                            TITLE not necessarily unique
69          123->4567                                       NAME not necessarily unique
                                                            ORGANIZATION does not uniquely determine DPHONE
                                                            DPHONE does not necessarily uniquely determine ORGANIZATION
70          1->23456                                        NAME not necessarily unique
71          1->23456789(10)(11)(12)(13)(14)(15)(16)         OFFICE does not uniquely determine DPHONE
             ->(17)(18)(19)(20)(21)(22)(23)(24)             POS_NO unique at AFIT level
                                                            DPHONE does not necessarily uniquely determine OFFICE
72          12->12
73          1->23456                                        SSAN not a key because empty positions are tracked too
74          1->23456789(10)(11)(12)(13)(14)(15)(16)         NAME not necessarily unique
             ->(17)(18)(19)(20)(21)(22)(23)(24)(25)         ED_CODE does not necessarily uniquely determine DPHONE
```

Appendix I,I

```
                ->(26)(27)(28)                                  ED_CODE does not necessarily uniquely determine
                                                                DUTY_LOCATION
                                                                ED_CODE and ENTRY_DATE do not necessarily uniquely
                                                                determine SECTION
75              13->2
76              1->23456789(10)(11)(12)(13)(14)(15)(16)(17)     NAME not necessarily unique
77              1234->5                                         AMOUNT may vary for a given PAYMENT_TYPE
78              12->12
79              1->23456789
80              123->456789
81              1->23456789(10)                                 TITLE not necessarily unique
                10->123456789                                   DTIC_NO is unique
82              1->2                                            LOCATION does not uniquely determine THESIS_SPONSOR
83              12->3456789(10)
84              1->23456789(10)
85              1->2
86              1->234
87              1->23456789(10)(11)(12)(13)(14)(15)(16)
                 ->(17)(18)(19)(20)(21)
88              12->3456789(10)
```

NOTES

1. X-> Y means the set of attributes X determines the set of attributes Y, e.g., 1->2 means attribute 1 functionally determines attribute 2. 2. Attribute numbers used in describing functional dependencies are based on the ordinal position of the attribute within a given relation, e.g., 1->2 means that attribute 1 is the first attribute in the relation and attribute 2 is the second. 3. 123478->569(10) means that the set of attributes {1,2,3,4,7,8} functionally determines the set of attributes {5,6,9,10}.

MISSING DATA ELEMENTS FOR:

AFIT/LS

STUDENT DATA: Place of Birth, Religion, Childrens names, GRE/GMAT scores (other than totals), projected assignment, GPA, Training report data, transcript (explicit)

FACULTY DATA: Dependent Names, Religion, Parking space assignment, PME data, Teaching hours

AWARDS: Criteria for selection, date award should be submitted

COURSE DATA: Course Outline

PROBATION STATUS: Name, Date Entered

END OF COURSE CRITIQUES: Name, Student Status (PCE or grad), Course Number & Offering, Date critique completed

END OF PROGRAM CRITIQUES: Name, Student Status, (PCE or grad), Date critique completed

SURVEY OF GRADUATES: Student Name, Duty Title/Assignment, Date survey completed, Name of Supervisor

SURVEY OF SUPERVISOR OF GRADUATES: (Same as SURVEY OF GRADUATES)

BUDGET: School, department, EEIC, FY allocation, Quarterly allocation

PCE STUDENT LOAD: Requirements, Quotas, Fill Rate

PART TIME STUDENTS: Students Program

SUSPENSE LIST: Date, Project due

OTHER GENERAL TYPES OF DATA REQUESTED BY AFIT/LS:

Faculty-Staff positions which are open.

General Registrar type data.

Recall Roster.

For all military faculty members, all MPC data which is not already acounted for in the design.

All the UMD data for the faculty and staff so they can "MANAGE THEIR VACENCIES", e.g. Name, Position Number, AFSC, Remarks

SF52 Data: WHEN PASSED TO: ED DATE:
WHEN PASSED TO: DET 10 DATE:
WHEN PASSED TO: CLASSIFICATION DATE:
WHEN PASSED TO: HIRING DATE:
WHEN PASSED TO: ACTIVITY FOR HIRING DATE:

OTHER IMPORTANT COMMENTS:

Colonel Smith expressed the following concerns:

1. Decide on our database management system fast. Try to get a free one so we don't get hung up in procurement.

2. Save us some fields so we can add data elements later.

3. Please don't centralize just for the sake of centralization. If we do, it will bog down and become useless because it is too hard to make changes. Keep it user friendly by making it user specific.

(NOTE: Data on items missing at the LS division level includes only that data which was not already listed above for the directorate.)

OFFICE SYMBOL: LSY

Expended TDY money as part of department budget.

Validated Advanced Academic Degree (AAD) positions in USAF by major command.

Validated Advanced Academic Degree (AAD) positions filled by compatible AAD holders.

Assignment of numbers of AFIT degree program graduates to AAD positions and to non-AAD positions.

OFFICE SYMBOL: LSM

Thesis abstact.

Manpower authorization data by grade, AFSC, FYQTR, position number.

Personnel data matched by position number, for both military and civilian.

Planning factors for TDY and budget estimation requirements (Common locations). Data required includes locations and airfares (both military and civilian), perdiem (both .ilitary and civilian), and average registration fees.

Additional student data: output assignment, including command and location, output AFSC, and duration of output assignment. (Tie to MPC required)

Requirements for advanced academic degrees. Capability required to obtain: a) number of academic degree requirements by speciality code b) number of graduates in the inventory possessing the degree code.

OFFICE SYMBOL: LSP

Contract regulation manuals.

OFFICE SYMBOL: LSB

Data on faculty consulting/research interest or activities (including a keyword search facility).

Data on faculty teaching interests and activities.

Compendium of people in AFIT Speakers Bureau.

Data on quality circles, e.g., literature references, consultants, contacts Current address for any student ever at AFIT.

Data from a proposed learning style questionnaire to be given to all incoming students.

Data from a teaching style questionnaire to be given to all faculty.

The data and the ability to isolate it for students in teleteach and candid classroom courses.

Data on all previous work experience of all students and staff (residents).

Thesarus of keywords for the database as a whole.

Data on student/faculty hobbies.

Student housing requirements.

Emergency phone pyramid.

MISSING DATA ELEMENTS FOR:
AFIT/EN

STUDENT DATA: Place of Birth

STAFF: Dependent names, Religion, all previous academic ranks, professional development data, honors (if not recorded in AWARDS), professional registration, civil organizations.

COURSE DATA: Course Outline

General Registrar type data

Past faculty course assignment data

FACULTY REPLACEMENTS: Name of person being replaced

(NOTE: Data listed as missing at the division level does not include data already listed at the EN directorate level)

OFFICE SYMBOL: ENA

Authorization figures for EN manpower report (Manning figures will be available in the database as designed, assuming that secretaries and technicians are considered to be staff and are distinguishable from each other, and are both entered into the STAFF relation.)

Maintenance cost data.

Telephone Control Report data.

For students and staff, data on the type of security investigation that has been completed on them, as well as their clearance level and the date their security clearance was completed.

Data on those students currently in Phd programs and scheduled for AFIT faculty positions.

AF Form 991 data.

AF Form 379 data.

AF Form 1098 data.

SF 971 data.

Data for tracking status of supplies, equipment, maintenance, reimbursements, and OA's.

OFFICE SYMBOL: ENP

IMPORTANT COMMENTS:

ENP would agree to storing data on their staff in the database only if they had total control over the data, including limiting access to the data only to ENP.

OFFICE SYMBOL: ENG

PUB DATA: page numbers

COURSE DATA: Classification

OFFICE SYMBOL: ENS

Data for training reports.

AFIT Form 89 data.

OFFICE SYMBOL: ENY

STUDENT DATA: Religion

DISSERTATION DATA

GRADES: Indicator on grade data telling that the given grade was for a repeat of the class.

COURSE DATA: ABET catagory

PAST COURSES: ABET Catagory

FOREIGN STUDENT DATA: TOEFL score

Consortium cross registration data

MISSING DATA ELEMENTS FOR: AFIT/LD

Ability to search thesis data using subject keywords.

PUB DATA: Report Number, AD Number, Volume, Issue, Pages.

BOOKS: Publisher, Edition, Price, ISBN number, Requester.

Recurring data for serial updates: Volume, Edition, Date (month published).

HOLDINGS: Author, Title, Subject, Copyright date, Call Number, Location (Bldg number), Volumes &/or Editions (as required). Includes titles of conference and symposium proceedings and series listings.

Circulation: Registration student name, card number, class, date registered, date registration expires. For non-students, date registered and date registration expires.

Data on Balance record of library acquisitions by: library, type books, type bound periodicals, theses.

Data on fund commitments by school and for: books/periodicals, deposit accounts, binding, computer services.

AF 971 data with training records including name, date, and subject. Also, attendance/time data.

Statistics on:

circulation - books, periodicals, reports
registration - total, new, mil, civ
interlibrary loan - sent, received
photocopies - sent, recieved, number of pages
microfiche - recieved, with number of fiche pages
reports received/totals
reports requested

MISSING DATA ELEMENTS FOR: AFIT/DA

STUDENT DATA: Place of Birth, Religion

FACULTY DATA: Dependent Names, Religion

COURSE DATA: Course Outline

POM DATA (5 year plans)

MISSING DATA ELEMENTS FOR: AFIT/PA

STUDENT DATA: Place of Birth, Religion

STAFF DATA: Dependent Names, Religion

MISSING DATA ELEMENTS FOR: AFIT/ED

Any teleteach/candid classroom data not included in present database design AFIT(CI & RR) input/output by month, by service and intra service breakdowns.

MISSING DATA ELEMENTS FOR: AFIT/ACB

(NOTE: This data was recieved from ACB under the title "ADP REQUIREMENTS". It is presented in its entirety in order to not distort it and some of it may included in the database design in some form already. The main reason much of this data was not included was because the form in which it was recieved did not lend itself readily to being broken down into the individual data elements required for the database. This area required an in-depth analysis of its own in order to determine the detailed requirements and data elements needed to fulfill them.)

Data on the following areas:

TDY (Student) LS, EN, DE

1.) School

A) Total class and individual student TDY costs.

B) Entry of estimates (EEIC's 407,408,409).

C) Entry of Actual Costs (EEIC's 407,408,409).

a) Comparison of estimates vs actual (over/under dollars).

b) Percentages of estimates vs actual.

D) Comparison of Prior Fiscal Years.

a) to dollars (actual and estimates).

E) Computation of course and travel days.

a) Develop cost of per diem per day - EEIC 409.

b) Develop Aug days (include travel days) of each course.

F) Project Fiscal Year costs based on program completions, costs and balance of fiscal year remaining.

TDY (Student) CI

1) Program (AECP, HPSP, etc.)

continue as above

TUITION CI (Costs and man-year computations)

1.) PROGRAM (AECP, HPSP, etc.)

A) School and course.

B) Dates of entry and completion.

a) No. of days, months attended for computation of man-years.

C) Costs of Tuition (by student & school).

a) Period of billing (Term &/or year).

b) Computation of fiscal year costs.

D) Computation of Man Year Costs/Entries.

a) STUDENT COSTS	STUDENT COSTS
MYs	Entries

E) Comparison of Prior Fiscal Years (2 years prior, current + 1 year forward).

a) Total costs by school.

b) Cost per student.

c) Cost per man year.

F) Project fiscal year costs, MYs by program based on completions, costs, and balance of fiscal year remaining.

C) Program estimates reimbursable to students for textbooks, supplies, etc.

a) identify payment totals and project balance to be paid based on MYs and entries by program.

BUDGET PROGRAM

1. Retrieval of data in 2750 ABW/b3500 system.
2. Development of budget by fiscal year.
   a) By PE code, EEIC, and RC/CCs.
   b) Compute estimates based on bogey, prior year requirements and future year program changes.
   c) develop distribution of funds based on annual/quarterly authority available.
      1) track unfunded requirements.
3. Track AFIT budget through fiscal year.
   a) Compare programs to obligations.
      1) Develope rates/trends.
      2) Project future/EOY costs for current FY.

REPORT TO THE AFIT COMPUTER CONFIGURATION BOARD

on the

CONSOLIDATED AFIT DATABASE and INFORMATION SYSTEM (CADIS)

by

Capt. Jeffrey S. Ricks

and

Lt. Robert Colburn

19 November 1982

7 ATCH
1. Database Diagram
2. Relation Descriptions
3. Relations & Attributes
4. Attribute Descriptions
5. Database Sizing estimates
6. Missing Information
7. DBMS Rankings

INTRODUCTION

This report contains the final results of the analysis of the AFIT information and data requirements survey and analysis plus an evaluation of commercially available Data Base Management Systems (DBMS). For additional background material or more detailed explanations, the reader is referred to the complete AFIT/EN thesis by Capt. Ricks and Lt. Colburn concerning the Consolidated AFIT Database and Information System (CADIS). This report is an extract of that thesis and presents only the results of the project. Subjects such as the general capabilities and functions of a DBMS, the reasons behind a centralized database and a detailed explanation of the DBMS evaluation process are covered in detail. Copies may be obtained by contacting Maj. Charles Lillie, AFIT/EN, x52024.

RESULTS AND CONCLUSIONS OF THE AFIT DATA ANALYSIS

The process of conducting an in-depth requirements analysis has provided a great deal of data and knowledge about both the overall and specific information requirements of AFIT. In order to adequately present this enormous amount of information, gathered from interviewing approximately 40 people over a four month period, the data has been grouped into seven general categories. These categories include the basic assumptions and ground rules which governed the interviews, the actual results of the interviews, any unidentified or incomplete data requirements, recommendations for the management and organization of the Consolidated AFIT Database and Information System (CADIS), an implementation plan, and possible interfaces to CADIS.

General Assumptions of the Data Analysis

The basic assumption is that the previous thesis by Lt. Allred would serve as a basis or starting point. By using his results it would be possible to gain a great deal of insight into the nature of the requirements for the remainder of AFIT. Another assumption, and one perhaps just as basic, was that a single consolidated and integrated database would best serve the needs of AFIT. This does however, include a distributed database, or one whose data resides on more than one computer, but speciffically excludes multiple, independant database systems.

The general guidelines adopted in reqards to the overall conduct of the requirements survey and analysis were designed so as not to impede or restrict the type or nature of the input. Primarily, all requirements were considered to be valid. Users were not questioned as to why a particular piece of information was needed, only how it would be used. Therefore, all requirements were considered valid for possible inclusion the database design. This ultimately resulted in an analysis which was of a greater depth than anticipated and more inclusive than any done previously.

The other procedural guideline which was used also concerned the data requirements. Requirements which were incomplete, not clearly defined or beyond the scope of the final project would not be included in the database design. However, in the interest of completeness and accuracy, all such requirements are listed seperately, along with who identified it and an explanation as to why it was not included.

General Comments on the Interviews

The reception by the vast majority of people contacted in regards to this project was enthusiastic and supportive. However, most expressed frustration and dissatisfaction over the inability to maintain and effectively use the information required by their organization. Current functions and responsibilities, plus current and projected requirements and problems were all discussed freely and openly in a professional and receptive manner.

Almost universally, it was expressed that CADIS was desperately needed and that once implemented, would go a long way toward alleviating some critical problems, assuming certain principles were kept in mind. Foremost of these was that such a system must provide easy access to the database plus the ability to conveniently query and manipulate the data as need be. Applications programs and their subsequent support was generally not desired because an adequate DBMS could give them greater flexibility through the query facility and report generator without the cost of software development and maintenance. When discussing the need for reports and other required documents, it was found, as expected, that if proper facilities were present, the need for many paper records and inter departmental reports would vanish.

The School of Engineering (AFIT/EN) and the School of Systems and Logistics (AFIT/LS) were interviewed in greater depth than other areas because some of the basic data elements used were a result of the previous in-depth analysis.

It was felt that the individual department heads should be contacted in order to revalidate their overall requirements. In most cases this resulted in a great deal of new information as well as modifications to others.

Overall Data Requirements

The overall goal of this project was to identify all the data requirements and design an integrated database which would reflect all the current needs of AFIT. In reality however, some directorates presented data which they did not elaborate upon nor was there time to define completely. .Therefore what was actually gathered was a solid core of highly common data, which if implemented in a timely manner, should satisfy a very large amount of AFIT's information requirements. This reasoning is based on the fact that almost all of the requirements expressed revolved around a sub set of the overall data. This "central core" consists of general student, faculty and staff data plus some single relations which provide necessary and more detailed additional information.

Another goal of those interviews was to gather a list if all the reports and documents which were used within AFIT. However, this has already been completed by AFIT/ACD along with data flow diagrams for the entire organization. These may be found in the 1981 AFIT ADP Master Plan .

The overall structure of the database can be found in the form of a diagram in attachment 1. This chart represents each of the relations identified to date plus a depiction of the most important logical associations between them. A line connecting two relations means that there is at least one common attribute between them and that there is also a meaningful logical connection between the two. All student and staff data is accessed by the Social Security Account Number (SSAN) of the person of interest.

Many attributes in these relations are stored as codes or short names, but a user can obtain full titles of expanded information on most of these codes or abbreviated forms. This is accomplished by using a logical connecting code and retrieving the full description, or in some cases by referring to the data dictionary.

The chart in attachment 1 denotes no other meaning than that described above. The size of a figure, nor the length, nor the number of lines implies any information.

More information on the content of the database can be found in attachments 2, 3, 4 and 5. the first four essentially comprise a highly detailed data dictionary, while the fifth documents the database size estimates.

Attachment 2 gives a functional description of each of the relations on the chart along with the length of each tuple (record). Attachment 3 is another list of relations but this one shows the attributes which comprise each relation and the key attributes (elements) for each, denoted with an "*". Attachment 4 provides a detailed breakdown of all the attributes by listing the size of each attribute its description, an example of how it is to be used, a list of the relations which incorporate it, and a list of any synonyms it might have. Attachment 5 gives, for each relation, the estimated total number of records stored in memory at the end of the first years operation of CADIS. Also given are the factors going into each estimate and the major source of those factors.

Database Size Estimation

The estimated size of the database is approximately 28M bytes of data. This figure represents the total amount of memory required for one full year of operation (several relations maintain four quarters worth of data). No student history, other than a short educational history for 1000 students, or registrar data has been included in the estimate.

This estimated figure was derived by taking the size of each relation, as determined by the sum of the lengths of its attributes, and multiplying it by the estimated number of tuples for that relation. Estimates for the lengths of each relation were derived by actual counts, estimates from faculty or staff members or by author's best estimation. These numbers may be found in attachment 5.

For ease of computation it was assumed that all data would be stored in character format and that one character would occupy one byte of memory. This estimate does not include the core memory required for the DBMS itself to function.

Missing Information and Data Elements

The list at attachment 6, along with the data dictionary, encompasses the entirety of the AFIT information/data requirements identified during the interviews. Implementation of the database as designed will alleviate approximately 90-95% of the problems within AFIT. The missing data was compiled from two entirely different sources, the worksheet used during each interview and comments by the interviewee, and represents which could not be included into the design for the reasons listed below.

The reason this data was not included is because it was: a) submitted too late in the design process, b) stated in a format which was much too general in nature, or c) various elements were very specific to a single area and not desired by anyone else.

Because this was a thesis project, the largest single influencing factor was that of time. Each interviewee was contacted in sufficient time to respond with their requirements and have them included (some were given much more time that were others). However, there were those who were unable to identify their total requirements quickly enough. therefore, only that portion recieved early enough was able to be included. In general, the data which falls into this category is not to be considered "essential" and the current structure will fulfill the vast majority of the requirements.

Each person interviewed was asked to be as specific as possible as to what data was required for their area. Everyone was asked to use the worksheet to mark their input and any item which was not SPECIFICALLY identified as UNDESIRABLE as included. In spite of this, some people simply identified their "general ADP/information requirements" in a very general and non specific manner. For instance, the most common area identified like this was "budget data". Because of the nature of this thesis, there was insufficient time to perform the in-depth system analysis required to translate this type of request into actual database requirements. AFIT will have to supply the qualified personnel to work with these areas and perform this lengthy analysis at a later date.

finally, most areas supplied data which was rather narrow in use and was oriented to a small subset of the directorate (mainly the AFIT/EN and LS directorates). This type of data was excluded because it was felt that because they were not really "common", and should be reviewed for desirability and applicability across the directorate prior to their inclusion.

RECOMMENDED CADIS MANAGEMENT STRUCTURE

Volumes and volumes of material have been published over the years on the technical aspects of DBMSs. On the other hand comparatively little has been written on the subject of management techniques, practices or methods associated with implementing and running such systems. According to what literature is available it seems to fall upon the individual organization to review their situation and develop their own methods. There is however, a set of sound, concrete reasons for going to the trouble of developing such a structure. "The database will have a major impact on the enterprise, not only in terms of benefits but in terms of problems. Its fundamental aspect of the sharing of data among users and of greater central control of data are just the type of changes that can have severe political repercussions within an organization." (REF 2, p 5-1)

In order for an integrated database to be effective, such things as the setting of standard definitions, formats and usages for all data elements plus the associated monitoring and enforcement of these standards are essential. This will insure the existence of accurate and consistent data for all users who will quickly come to depend heavily upon the database for meeting their mission requirements.

"Resentment to the disciplines imposed by the database can often arise; some departments might insist on applying their own standards [or methods of database management]. Such difficulties can be overcome, but only if the continued and dedicated support [and guidance] of senior management is assured."

(REF 2, p 5-1)

The solution most often adopted, and recommended by industry practice and writing, is to establish a specialized area of the organization whose primary consideration is the management and control of the database and its operation. This 'manager' must be "familiar with the use and nature of all the data, ...be vitally concerned with the performance of the system and...be able to negotiate with and resolve conflicts between user departments..." (REF 2, p 5-24). In short this position should be filled with someone who is totally competent in all the technical aspects of a DBMS plus have intimate knowledge of the organization and its operation. But most importantly, he must be someone who has the authority to make and enforce decisions involving the database.

AFIT Management Features & Problems

In most respects AFIT is no different than other civilian and government organizations. There is a shortage of qualified personnel and those who are available do not have the necessary database experience to implement and control such a project. For the most part AFIT will cope with these universal problems just like everyone else and rely on the available pool of expertise from other areas and accomplish the project anyhow. There are some problems which while they are not unique to AFIT, present some real obstacles.

There is a decided lack of practical experience with the design, implementation and management of DBMSs. This, coupled with the fact that this would be a "new start" project could combine into a deadly combination if the proper precautions are not taken immediately.

AFIT has a diverse community of users which view the concepts of database systems and data automation from three extremely differing perspectives; that of academic, administration and real world. The result is a situation not too uncommon, that is, differing opinions on the control of data and the methods and procedures for implementing a centralized database.

CADIS Management Recommendations

The optimum solution for the administration of CADIS appears to be one with two components. First, "the solution most frequently offered for this type of situation is placing the administrative responsibilities in a staff position, reporting directly to the highest manager responsible for data processing in the organization." (REF 1, p 6) Secondly, "the database administrator functions should be a team rather than a single manager" (REF 2, p 5-24)

The Database Administrator (DBA) position is a very demanding one. He is faced with deciding such things as "resolving conflicts of ownership if several departments wish to amend or retrieve the same data...negotiating with users to establish who has the right to access or change data items...provide recovery facilities and establish precations for applications in the event of a breakdown...establish data entry, edit and validation standards, and maintain the data dictionary." (REF 2, p 5-26,27) "For the data administration to have sufficient authority to be accepted by user departments, to reconcile their conflicting requirements for data, and to be able to enforce standards, the position must be relatively senior within the enterprise." (REF 2, p 5-?)

Based on the above comments it is recommended that AFIT initiate two seperate levels or areas of administration for CADIS. The first, that of the Database Administrator (DBA), should be a senior member of the staff with sufficient authority and knowledge of the organization to shepherd the database, with all of its growing pains through to full adulthood.

To aid the DBA, and as part of the first level of CADIS administration, each directorate should have a single representative or Data Administrator (DA) which would represent the views and needs of their respective functional areas. Requirements and problems would first be reviewed by a DA for clarity, consistency and practicality. He should also be able to answer any question about current data usage and meaning as well as act as the focal point for any problems or changes to the database originating from or directed to his directorate. Once discussed at this level, anything of interest to the community of users, valid changes to the structure of the database or disputes should be taken to the DBA for review and action. The DBA would check them for validity, then for consistency against prescribed rules and standards and act accordingly.

Once the DBA has approved the changes or resolved any problems with the recommendations, they would then be passed on to the other component of CADIS management, the Database Manager (DBM). This person or group would act only on the direction of the DBA and should not be a member of the staff but someone with a more technical orientation toward DBMSs. This must be the case because he is the one who will monitor and be in charge of the DBMS software, make any changes to the structure of the database as dictated by the DBA, as well as being the interface to the vendor and acting as the ultimate in-house technical representative. The DBM should not be involved in data oriented problems, but only in the efficient operation of the DBMS itself on whatever computer system it is installed.

It is very likely that the CCB could suffice for the first level of the previously mentioned structure, with one member appointed to be the DBA, and the directorate representatives , or their appointees, acting as the DAs. The second level of administration could be performed by AFIT/ACD. They could supply one or more people to act as the DBM since the type of the technical expertise desired resides with them. The job of DBM would very likely be a full time assignment during the infancy of CADIS but could become a background duty once it is operational.

In general; each functional area needs to be responsible for their own data, AFIT/ACD should be in charge of the computer and the DBMS software and serve as technical representative for their combined use.

For this to become a viable management framework, the general level of education in regard to DBMSs has to first be increased. AFIT should immediately begin to send people to the database courses (basic and advanced) offered within the School of Engineering. They also need to talk to vendors and visit other AF installations which have experience in database systems. Additionally, the heads of each of the directorates should at least attend the database short course available periodically at AFIT/EN. These suggestions are wholeheartedly recommended because the uniqueness and nuances of DBMSs dictate a need to learn both the theory and practices of a database operation. Without this basic understanding of the functions and capabilities of a DBMS, by all those involved, it is nearly impossible to successfully implement CADIS and have it be as useful as it could be. Such education should be above and beyond the normal training classes which will be offered for the DBMS, usually by the vendor, once it is installed. The DBA and the DBM should be the first to seek this type of knowledge, and as soon as possible. They would then be followed closely by the DAs and their users. One can never hope to build and operate that which he does not understand and trust.

Database Implementation Recommendations

In order to implement a database and bring the users 'on-line' with the minimum amount of mayhem and confusion, plans must be drawn up for the implementation well in advance of the actual event. The purpose of this section is to suggest one scheme which would ease the 'labor pains'. A basic premise for any implementation plan is to try and achieve the most results with the least expenditure of and resources.

First, a Database Implementation Team (DBIT), composed of at least the DBA and DBM, should evaluate the user community in order to determine any highly critical areas which were not known at the time of the requirements analysis and superimpose them upon the following general plan discussed below. AFIT/ACD will have to supply three people on a full time basis for about a year in order to implement CADIS. After this time, the number could be reduced to two and eventually to one full time and one part time.

General CADIS Implementation

The DBIT should not under any circumstances attempt to load all the relations prior to allowing the users access to the database. This is just the common sense, and good top-down implementation practice which will prevent massive confusion and eliminate wasted human and computer time spent resolving problems. If all the relations are loaded at once, this would cause the majority of the users, who could operate with a subset of the data, to wait an unnecessarily long time before becoming operational. Also, by giving the data to the users in usable segments, they will have the opportunity to fully evaluate and test the features of the system and find any errors in the data or its structure easier than if the whole of their data was present. When a new system is presented, "complete" or "fully capable", the users tend not to learn it thouroughly nor learn how to use its full capabilities, but learn just enough to get by.

The inclusion of the registrar should be left aside at this time because of their current automation effort and the many complexities and controversies associated with this type of data. But if, at some later date when the all of CADIS has been implemented and exercised, it is deemed desirable, then and only then should AFIT/RRs inclusion be considered. What should be implemented at this time is the subset of the registrars data which was requested by the remainder of AFIT. By so doing it is assured that CADIS will be truly useful to its users.

Also, the mistakes and hardships which surround an organization trying to operate with its essential data needlessly spread among multiple data systems are eliminated. Finally, it would be easier all around, plus generate experience with the new system if each user was generally responsible for loading their own data. No one knows the data like the individual departmentmental users do, and with the DBA and DBM assisting by supplying routines and advice, implementation would proceed much faster.

CADIS Implementation Plan

The intent of this plan is to install or expand (load and verify) the database in sections which will do the most good for the most people. The steps are not necessarily mutually exclusive and several steps may be in various stages of execution at any given time. More capable users may be able to get their data loaded quicker than others and no attempt should be made to slow them down if they can be administered adequately.

The following steps consitute the CADIS Implementation Plan:

(1) The DBA and DAs should mutually establish the data standards and codes which will be used throughout the system.

(2) Define the entire database structure in the computer at one time. This would consist of those relations and attributes, along with their rules, as defined in the attachments or by the DBA. By doing this first, modification of the basic structure can be kept at a minimum and nothing extra needs to be done when a user is ready to load a relation.

(3) Initially load those relations which comprise the 'core' of CADIS under the supervision of the DBA. This will allow AFIT/EN, LS, DE, PA, ED, and CI to go into limited operation very quickly and allow them to start testing out the structure and their data. This 'core' is comprised of the STUDENT and

STAFF relations plus those relations which allow them to function properly. These core relations are: OFFICES, ACADEMIC_TITLES; STUDENTS, STAFF ED_CODES, SECTION, FOREIGN_STUDENT_DATA, CURRENT_THESES, CI_DATA; DE_DATA; RESIDENT_DATA; MAJCOMS, SPOUSES, LOCATIONS (only the subset large enough to allow minimum operations), and AFIT_CALENDAR.

(4) Once the 'core' has been installed and verified for completeness, the users can begin to load the special purpose relations which pertain to their particular operation. These relations include: BOXES, LOCKERS, ED_HIST, PAST_COURSES, ED_PLANS, EXTRA_DUTIES, EXTRA_DUTY_ROSTER, DUTY_OFFICER, PROMOTION_STATS, QUOTAS, PCE_DATA, DEGREE_SOUGHT, DEGREES and all the faculty related relations.

(5) Complete the loading of the LOCATIONS relation with the remaining data required by AFIT/CI and others.

(6) Finish installing the relations which pertain to AFIT/CI. These relations are: CURRENT_CI_PROGRAMS, CI_PROGRAMS, INST_PAYMENTS, INST_POCS, SCHOOL/EWI_DATA, MMEP_DATA, MED_BOARD_CERTIFICATION, MED_TOURS and MED_PROG.

(7) Load the course, book and room data. These relations consist of: AFIT_COURSES, PREREQ, COURSE_BOOKS, BOOKS and ROOMS.

(8) Load the course scheduling and other data from the registrars system. This includes the following relations: COURSE_TIMES, STUD_SCHED, SHORT_COURSES, FAC_COURSES, SHORT_STUDENTS.

(9) Load the leftover relations as the data becomes available.

(10) Begin loading the THESES and STUDENT_HIST relations as appropriate.

If this type of approach is followed, an effective management structure is instituted and the people are properly educated, CADIS can be at least 85%

operational in about one year. Impacting on this estimate are the decisions as to the DBMS and the hardware and the speed with which the data is gathered and prepared for entry into the database. Preparation could begin immediately after the DBMS and hardware is decided upon by storing it on the machine to be used via tapes or a text editor. This data can then be easily prepared for final loading easier and at a later date.

Interfaces to CADIS

For purposes of this section, the term "interface" will mean any data which one data system may provide or recieve from another data system. There is no intention to imply a method or media for accomplishing the actual transfer.

The following are recommended interfaces to CADIS:

(1) Let the AF Standard Personnel System periodically supply data on the AF students and staff at AFIT. One problem identified by AFIT/DP was that the data maintained by the schools on their students is almost always more current than that maintained by them and the AF personnel system. By having AFIT/DP supply those attributes and allow them to be updated via the interface only, the only way a particular students record could be changed is by going through AFIT/DP and modifying their permanent record. Changes in the data would then show up in the database at the next update. The primary reasons for utilizing this interface would be to assure AFIT of the most current, official data on on a student. AFIT could therefore prevent their data from becoming different from his permanent record and visa versa. (NOTE: AFIT/DE currently recieves a periodic data tape from Randolph AFB.)

(2) Allow the registrar to supply data to CADIS for the following relations: COURSE_TIMES, AFIT_COURSES; STUD_SCHED, FAC_COURSES, PCE_DATA,

SHORT_COURSES, SHORT_STUDENTS and the degree which a student is seeking. While this data probably does not exist in a directly transferrable format, enough can be supplied and converted to guarantee consistency across AFIT.

(3) Let CADIS supply the registrar with the student grades at the end of each term. Instead of faculty members filling out grade sheets and passing them to AFIT/RR. Grades for students would simply be entered into CADIS once and the whole school could transferred in one easy step. Also, departments and advisors would have the past grades which they requested for each of their students.

(4) When the students' schedule is recieved from the registrar (via interface number two) and entered into CADIS, a check could be made against the students' educational plan (ED_PLANS relation) which would flag any deviation from that plan for the faculty advisor.

(5) When the COURSE_TIMES data is supplied to CADIS, that same data could be used to update the ROOM_SCHED relation for the current quarter.

RESULTS AND CONCLUSIONS OF THE DBMS EVALUATIONS

General Procedure

As much information about available DBMSs was gathered from a wide variety of sources as was possible. The most valuable was an article which appeared in a September 1981 issue of DATAMATION which presented a short synopsis of each of DBMS on the market. With this as a guide, further information was gathered as required from other sources, such as DATAPRO and articles in periodicals. Several weeks were devoted, in whole or part, to studying this data and talking to several vendors in order to gain a thourough understanding of the available DBMSs and what the current state-of-the-art is.

The evaluation was carried out by searching for answers a set of questions and assigning points according to pre-defined rules. When there was no data or insufficient data so that a firm determination could be made concerning a item, a value of zero was assigned. In some cases the vendor was contacted for clarification as well as additional detailed information on those DBMSs which scored a 70 or higher (the top 14 systems) on the evaluation was requested. The total values were calculated for each DBMS and they ranked accordingly (attachment 6). A chart showing how each DBMS scored in each category may be found in attachment 7.

General Comments on the Procedure

Across the board there is little specific, detailed information available to the public on DBMSs. There are several sources from which very general information may be obtained, but little of the type required by this evaluation. DATAPRO has a fairly detailed breakdown on DBMSs (including most of what was need for those systems listeded), but mainly on those systems oriented toward IBM and other larger computer systems. It appears that a large segment of the market has been overlooked and DATAPRO's descriptions exclude many "recent" and very capable systems. Sometimes that information which was available was ambiguous or misleading as to the actual capabilities and functions of the DBMS.

Another aspect which was difficult to pin down were the DBMSs which were available from within the Federal Government. The General Services Administration (GSA) Federal Software Exchange program either did not contain the desired information or simply did not list possible systems. One DBMS (RIM) which was written under a government contract, known by the author and others at AFIT, was evaluated but did not appear on the list. This most likely is a failure on the part of the organizations to list their systems and not on the system itself.

In spite of the lack of data in some areas and because of the way in which the evaluation was structured it is felt that those systems which scored the higher points are the most suitable for AFIT. It is felt, very strongly, that the evaluation was sufficiently encompassing and well rounded to adequately reflect the needs of AFIT. Additionally, as was hoped, the best possible DBMSs were identified through a fair and impartial process.

Conclusions

Several important conclusions were reached during the course of this evaluation. They fall into two general categories. One deals with AFIT itself and the other concerns the DBMS market.

Most important is the fact that the capabilities which AFIT requires of a DBMS are well within the current state-of-the-art in DBMS technology. The systems now being marketed are extremely sophisticated and written mainly for the non-programmer. They provide a great deal of flexibility for the user and the organization alike in both the manipulation of the data and the database structure itself. Additionally, they virtually all but eliminate the need for applications programs in organizations which have information requirements similar to AFIT. Extremely sophisticated, english like query facilities combined with good security measures, report generators and accounting capabilities provide a DBMS which can be virtually tailored to the individual users.

Two less significant facts were also discovered. As far as industry views databases and their uses, AFIT represents a rather small application. This is in spite of the potentially large amount of data to be housed in the database. Database applications are generally classified by the amount and frequency of queries levied against the data. Even at peak usage, AFIT will not amount to a significant level, as defined by community. Another thing learned was that AFIT has a rather unusual mixture of computer hardware on which to implement a DBMS. There are a limited number of DBMSs currently available which can run under the UNIX operating systems. However, those which are available are very powerful and popular. As few as these were, the number of DBMSs capable of running on the HARRIS minicomputer are fewer still. This situation severely limits the range of possible DBMS candidates for AFIT.

Because of the large number of DBMSs which were evaluated, some general observations may be made about the market as a whole. Primarily, there are many good DBMSs which will more than satisfy AFIT's requirements, and at reasonable price.

Secondly, the majority of the DBMS market appears to be oriented toward IBM and DEC hardware. This is not all to surprising because these two vendors manufacture the majority of hardware presently in use throughout the world. On the other hand, there does seem to be a trend by smaller software companies to fill the gap and provide capable and reliable DBMSs for a larger variety of hardware and operating systems. It is probably safe to state that there is at least one DBMS which will operate on almost every computer currently commercially available. This does not necessarily include microcomputer even though they too are being quickly accounted for.

Recommendations

Recommendations for the DBMS to be utilized within AFIT to support CADIS are based upon the current hardware configurations at AFIT, the information requirements and performance characteristics expressed in the interviews, and the current state of DBMS technology.

It was determined at the conclusion of the evaluation procedure that those systems which scored a 70 or higher were to be considered as prime candidates for implementation. These systems are the top 14 systems out of 44 evaluated. Cost figures are not included in the recommendations nor were they considered heavily because a specific price must be negotiated with the vendor and might very well be lower or higher than that quoted in the literature, depending on the individual vendor.

There are two almost equally, highly qualified DBMSs which should be used to implement CADIS, they are ORACLE and SEED. The capabilities and functions of each are almost identical in all respects. Each has an advanced level of data security, a data dictionary, fully capable report generators, database and data maintenance facilities and a highly sophisticated and useful CRT screen formatting facility for applications development.

Two of the most significant features of these DBMSs are the security and CRT screen formatter. Security is controlled not only with passwords and person-ids to the attribute (data element) level, but they allow for the logical data views to specify restrictions on a subset of data within a relation, such as STUDENTS. This means that each directorate can control the priveledges on their students, or other data, not only for themselves but for the rest of AFIT as well. As far as can be determined, these are the only DBMSs currently available which offer such extensive and flexible security measures (others merely allow password and/or person-id protection).

Almost as significant as the strong security is the existence of an easy to use (user oriented) and very capable CRT screen formatter. This feature could all but eliminate the need for applications programs at AFIT, including those required for AFIT/CI. With the use of any CRT which allows for direct cursor addressing, a user may designate protected and unprotected areas for data entry, specify fields which cannot be left empty or blank, specify special edit checks on the data (this includes having the DBMS check the database to see if a value is already present in the database; especially useful for codes). In addition, the prompts and error messages the user sees can be specified as well as special on-line , application dependent, help files. The following alternatives are strongly recommended for the implementation of CADIS.

(1) Acquire the ORACLE DBMS, currently on the GSA price schedule, and implement it on the PDP 11/34, PDP 11/60 or VAX 11/780 under the UNIX operating system (NOTE: ORACLE may be selected fcr operation on the HARRIS at a later date.)

(2) Acquire the SEED DBMS, currently on the GSA price schedule, and either remove UNIX from one of the PDPs and reinstall the original DEC (VMS) operating system (SEED does not yet run under UNIX), acquire another DEC computer system or some other which will run SEED, or negotiate with the vendor to make SEED operate on the HARRIS (supposedly projected for accomplishment by the vendor in the future).

Both ORACLE and SEED scored very high in our evaluation (attachment 7) and, after recieving more detailed literature from the vendors, it seems that they are truly effective, easy to use and rather popular and are in wide use throughout the data processing and business community. ORACLE has received high marks from the Strategic Air Command and is strongly being considered by the Naval Post Graduate School. SEED was chosen for and is highly recommended by a consortium of Canadian universities and the US NAVY Weapons Development Center. Both vendors offer training and assistance in establishing a database and are available at almost any hour to answer questions (SEED maintains a 24 hr hotline) or solve problems. It is highly recommended that the DBMS which will support CADIS be one of these two systems.

Another benefit for AFIT may be the fact that both vendors alluded to a substantial cost reduction if their system could also be made available for "academic" uses. By allowing students and faculty to access the DBMS and use it for their own learning purposes, AFIT might qualify for a special "university" purchase price.

BIBLIOGRAPHY FOR THE CCB REPORT

(1) Aeurbach, "Guide to Data Base Management", *Auerbach Publishers Inc*, Philadelphia PA, 1975.

(2) Palmer, I., *Data Base Systems: A Practical Reference*, QED Information Sciences Inc, Wellesley MA, 1982.

DATABASE MANAGEMENT SYSTEMS EVALUATION SHEET

	DBMS	VALUE	REMARKS
HIGH PRIORITY CONSIDERATIONS Rating Scale: (Max =10)			
1. Current Availability			
2. Structure			
3. AFIT Computer Systems Compatability			
4. Memory			
5. Cost			
6. Machine Independence			
7. Data Dictionary			
IMPORTANT OPERATIONAL FEATURES/CONSIDERATIONS Rating Scale: (Max=8)			
8. Applications Languages			
9. Privacy & Security			
10. Backup & Recovery			
11. Structure Definition			
12. Modifiability of the Structure			
13. Maintenance			
14. User Friendly			
15. Resources (optional/required)			
16. Update/Retrieval Protocol			
17. Report Generator			
18. Vendor Support			
19. Documentation			

DESIRABLE FEATURES
Rating Scale: (Max=6)

20. Batch/On-line Operation

21. Utilities/Functions

22. Accounting Package

23. Field Value Enforcement

24. Key Enforcement

25. Secondary Indexing

26. User Usage Statistics

LIST OF DBMSs ACCORDING TO THEIR EVALUATED SCORE
MAXIMUM SCORE WAS 192 POINTS

DBMS	VENDOR	SCORE
SEED	INTERNATIONAL DB SYSTEMS	177
ORACLE	RELATIONAL SOFTWARE INC	169
IDMS	CULINANE	154
RIM	BOEING COMPUTER CO	103
TOTAL	CINCOM	101
INFO	HENCO Inc.	97
INGRES	RELATIONAL TECHNOLOGY	91
DRS/XSB	ARAP ADVANCED DATA MGMT DIV	82
SQL/DS	IBM	90
MODEL 204 DBMS	COMPUTER CORP OF AMERICA	80
SYSTEM 38	IBM	78
ADABAS	SOFTWARE AG OF NORTH AMERICA	78
SYSTEM/2000, 2000/80	INTEL CORP	73
DATACOM/DB	APPLIED DATA RESEARCH	70
DPL	NATIONAL INFO SYSTEMS	63
INQUIRE	INFODATA SYSTEMS INC	63
SYSTEM 1022	SOFTWARE HOUSE	60
DBMS-10/20, DBMS-II	DEC	59
RAMIS II	MATHEMATICS PRODUCTS GROUP	59
RAPPORT	LOGICA INC	57
IMAGE/3000	HEWLETT-PACKARD	57
DMS-90, DMS-1100	SPERRY-UNIVAC	53
DMS-170	CONTROL DATA	51

DM-IV	HONEYWELL	44
IMS, IMS/VS	IBM	44
UNIBASE	RLG CORP	43
CREATE/3000	CRI INC	36
DL1/1, DOS/VS	IBM	35
MADMAN	GENERAL ELECTRIC CORP	32
DBMS PRIME	PRIME COMPUTER INC	32
DBMS-990	TEXAS INSTRUMENTS	31
CREATE	COMPLETE COMPUTER SYSTEMS	30
IDS I/II	HONEYWELL	27
SIBAS	SHIPPING RESEARCH SERVICE	23
SIR	SCIENTIFIC INFO RETRIEVAL INC	22
IDOL	SCIENCE MANAGEMENT CORP	20
DAI	CONSOLIDATED BUSINESS SYST	19
DG/DMS	DATA GENERAL	19
SUPERSETUP	THE AUTOMATED QUILL	18
AMBASE	AMCOR COMPUTER CORP	16
SYSTEM C	SOFTWARE CLEARING HOUSE	16
DIRECT	BANCHOHIO CORP	15
MINDS	MINI DATA SYSTEMS	15
DATABASE MANAGER 1	CONDOR COMPUTER CORP	12

DATABASE EVALUATION MATRIX

	DBMS	EVALUATION FACTORS 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1.	ADABAS	2	6	1	3	?	9	10	5	4	6	0	0	0	8	0	0	6	0	0	6	3	0	0	0	3	6	78
2.	AMBAS	2	0	0	0	?	8	0	0	3	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	16
3.	CREATE	2	0	0	3	18k	6	0	0	2	0	0	0	0	6	0	0	6	1	1	0	1	0	0	0	2	0	30
4.	CREATE/3000	2	10	0	3	10k	5	0	4	0	0	0	0	0	3	0	0	0	1	1	0	1	0	0	0	6	0	36
5.	DAI	2	0	0	3	?	5	0	2	0	6	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	19
6.	DATABASE MANAGER	2	0	1	3	10k	1	0	4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	12
7.	DATACOM/DB	2	10	0	3	57k	4	9	8	3	6	0	0	0	1	0	4	0	3	0	6	2	0	0	0	3	6	70
8.	DBMS PRIME	2	1	0	7	34k	5	0	4	2	2	0	0	0	1	0	2	0	0	0	6	0	0	0	0	0	0	32
9.	DBMS-10/20 DBMS-11	2	7	1	3	?	5	5	8	7	6	0	0	0	0	0	0	2	0	0	6	2	0	3	0	0	2	59
10.	DBMS-990	2	0	0	3	?	1	5	6	2	3	0	0	0	1	0	0	0	0	0	6	2	0	0	0	0	0	31
11.	DC/DMS	2	1	0	3	?	1	0	2	0	2	0	0	0	0	0	2	0	0	0	6	0	0	0	0	0	0	19
12.	DIRECT	2	0	0	3	14k	4	0	2	0	0	0	0	0	1	0	2	0	1	0	0	0	0	0	0	0	0	15
13.	DLT/1 DOS/VS	2	2	0	3	434 mo.	4	5	4	2	3	0	0	0	0	0	0	2	0	0	1	1	0	0	0	0	6	35
14.	DM-IV	2	7	0	3	*1	5	1	4	3	3	0	0	0	0	0	0	2	0	0	6	2	0	0	0	0	6	44
15.	DMS-170	2	10	1	3	30k	5	5	5	3	1	0	0	0	0	0	2	2	0	0	6	0	0	0	0	0	6	51
16.	DMS-90 DMS-1100	2	7	0	3	750 mo.	4	5	5	6	3	0	0	0	0	0	2	3	0	0	6	1	0	0	0	0	6	53
17.	DPL	2	10	1	3	38k	10	5	4	3	4	0	0	0	2	0	2	2	0	0	6	2	0	1	0	0	6	63
18.	DRS/XSR	2	10	3	3	60k	10	10	5	8	6	0	0	0	2	2	2	3	0	0	6	4	0	0	0	0	6	82
19.	IDMS	3	8	0	10	*2	5	10	8	8	8	6	7	6	8	4	8	6	8	5	6	6	6	0	6	6	6	154
20.	IDOL	2	0	0	3	4k	9	2	0	0	0	0	0	0	2	0	0	1	0	0	0	1	0	0	0	0	0	20
21.	IDS I/II	2	1	0	3	400 mo.	3	2	2	2	3	1	0	0	0	0	2	0	0	0	6	0	0	0	0	0	0	27
22.	IMAGE/3000	2	7	0	3	5k	9	5	5	4	7	0	1	0	2	0	2	1	0	0	6	3	0	0	0	0	0	57
23.	IMS, IMS/VS	2	2	0	3	1k mo.	5	6	3	2	2	1	0	0	0	0	2	2	0	0	6	2	0	0	0	0	6	44
24.	INFO	2	10	0	10	?	2	10	0	8	0	6	6	4	7	0	0	6	0	8	4	5	3	6	0	0	0	97
25.	INGRESS	2	10	1	7	*3	2	8	5	2	2	6	4	2	4	0	4	4	4	0	4	6	0	2	0	6	6	91
26.	INQUIRE	2	9	0	7	*4	5	6	5	6	4	0	0	0	2	0	2	0	0	0	6	3	0	0	0	0	6	63
27.	MADMAN	2	1	1	4	20k	5	0	0	2	4	0	0	0	0	0	3	0	0	0	2	2	0	0	0	0	6	32
28.	MINDS	2	1	0	3	5k	4	0	0	0	0	0	0	0	1	0	0	1	0	0	2	1	0	0	0	0	0	15
29.	MODEL 204 DBMS	2	10	0	3	*5	10	5	5	5	3	2	4	0	4	0	3	3	0	0	6	3	0	0	0	6	6	80
30.	ORACLE	3	10	10	10	?	10	10	8	8	8	6	8	6	8	2	6	8	7	6	6	6	3	5	6	6	3	169
31.	RAMIS II	2	10	0	3	32k	8	5	5	2	2	0	0	0	2	0	2	4	0	0	6	2	0	0	0	0	6	59
32.	RAPPORT	2	10	10	3	*6	10	0	8	2	8	0	0	0	0	0	2	0	0	0	0	2	0	0	0	0	0	57
33.	RIM	10	10	1	7	8k	2	10	4	6	0	6	6	0	8	0	0	3	6	?	4	4	0	5	5	6	0	103
34.	SPED	3	10	9	10	?	10	10	8	8	8	6	8	6	8	2	6	8	8	8	6	6	6	5	6	6	6	177
35.	SIBAS	2	1	2	3	40k	6	0	6	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23
36.	SIR	2	7	1	3	5k	6	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	22
37.	SQL/DS	2	10	0	7	*7	4	10	3	4	3	4	7	1	3	0	4	3	4	0	4	2	0	0	6	6	3	90
38.	SUPERSETUP	2	0	0	6	*8	3	0	0	1	0	0	1	0	1	0	2	0	0	0	0	2	0	0	0	0	0	18
39.	SYSTEM 1022	2	10	0	3	*9	3	0	4	2	8	1	0	0	0	0	2	5	0	0	6	2	0	0	0	6	6	60
40.	SYSTEM 38	2	10	0	5	?	1	10	3	6	2	3	2	4	6	0	3	3	3	4	6	5	0	0	0	0	0	78
41.	SYSTEM C	2	0	0	3	19k	2	0	0	0	0	0	0	0	0	0	8	0	0	0	0	1	0	0	0	0	0	16
42.	SYSTEM/2000 2000/80	2	7	1	6	70k	10	10	5	4	4	0	0	0	1	0	4	5	0	0	6	2	0	0	0	0	6	73
43.	TOTAL	10	5	10	7	?	10	10	6	2	4	2	0	4	3	0	2	0	4	0	6	4	0	0	6	0	6	101
44.	UNIBASE	2	1	6	3	25k	5	0	1	0	0	0	0	4	0	0	2	5	0	0	6	2	0	0	0	6	0	43
MAX VALUE		10	10	10	9		10	10	8	8	8	6	8	6	8	8	6	8	8	6	6	6	6	5	6	6	6	192

NOTES: 1. 2500 mo. 2. 50k+ 3. 30k, 2k maintenance 4. 120k 5. 170k 6. 12-44k 7. 300 mo.,105 mo.
8. 2.5k 9. 17-54K ? = data not available

Appendix IIC

DATA ANALYSIS QUESTIONNAIRE FOR THE AUTOMATION OF THE INFORMATION REQUIREMENTS OF AFIT/CI

The purpose of this questionnaire is to gather information pertaining to the establishment of the AFIT/CI portion of CADIS and the development of the required applications programs. Responses to these questions should be formulated in such a way as to provide sufficient data to aid in the development of your system. The accuracy and completeness of both the database and the programs which access it will be directly influenced by your responses.

(1) The section deals with your current situation as well as with organizational and functional data you now use.

A. What is the organizational structure of AFIT/CI and where does it fit within AFIT? Briefly describe the function of each level.

B. What programs are managed by AFIT/CI and which functional area is responsible for each?

C. How do each of the programs managed by AFIT/CI interrelate to one another?

D. What items of data does each functional area require to perform its job? (break them down by AFIT/CI program)

E. For each data element, what is the proper/complete name, size and valid values?

F. List all the known relationships between the data elements.

(2) This section deals with how the data elements discussed above are used. The questions deal with data element usage and general information requirements which use them.

A. The two broad categories of data used are specified as Financial and Personal data. Further define each of this division, also discuss functions and programs which relate to each.

1. On a routine basis, what information (output)

from each of the categories outlined above is required? (what questions will need to be routinely answered; what reports and displays routinely generated?)

2. List any possible ad-hoc or unusual questions you have encountered in the past or envision might be asked.

B. For each of the categories discussed in 2.A:

1. List the data elements which will be required to fulfill each of the queries you listed.

2. What is the format in which you might like to have the answers to each query appear?

3. What is the format of each of the reports?

(3) Privacy and Security requirements for CADIS.

A. Who is now authorized access to the data and who will be authorized use of the various relations in CADIS? (break down by AFIT/CI program)

B. What restrictions/authorizations will need to be imposed to control access to the database for daily use, dissemination, review, etc?

C. What types of audit trails or change controls are currently in effect? What will have to be implemented within CADIS?

D. How detailed must these audit/check procedures be (by person, time, functional area, etc.)?

(4) This section deals with the overall capabilities and functions of CADIS.

A. What is the skill and educational level of the personnel who will be using CADIS?

B. Will the same people perform input, change, deletion and query functions? If not, who will do what

C. How often will the system be used for:

1. Data input?
2. Data changes?
3. Queries?

D. What new functions will have to be incorporated into CADIS and the application programs? (over and above present capabilities?)

E. Who will act as the functional manager of the ZEIS/CI portion of CADIS? The programs? The data?

F. When must the system be available? What would be a desired average time to restore the system in the event of a failure? (communications, hardware and software)

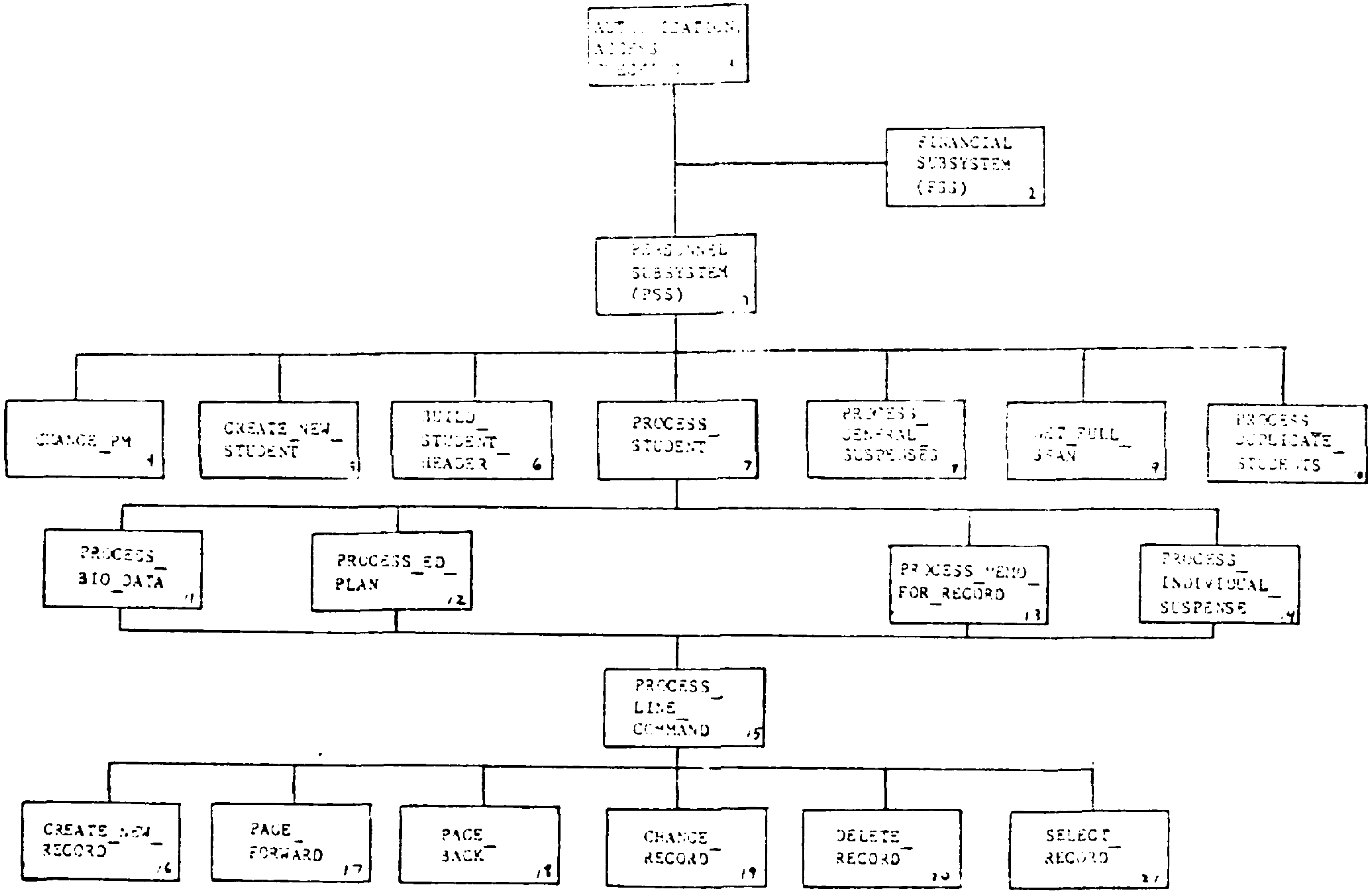
G. Are there any time periods which will require a significant increase in the use of the system?

H. What kind and degree of archive data will have to be retained? For how long? How will it be used?

I. What range of turnaround time is desired for routine queries? Ad-Hoc queries? Reports?

J. What are the backup and recovery requirements or desires for system reliability and integrity?

APPENDIX IIIB



PERSONNEL SUBSYSTEM SOFTWARE (PSS) HIERARCHY CHART

A FUNCTIONAL DESCRIPTION OF THE

AFIT/CIR/M PERSONNEL SUBSYSTEM SOFTWARE (PSS)

The following is a description of the AFIT/CIR/M Personnel Subsystem Software (PSS) from the user's viewpoint. PSS is designed to be an extremely user friendly interface between CADIS and AFIT/CIR/M. Since PSS is menu driven, the description will focus on the various menus (screens) that the user will see when utilizing the system. For any menu which requests the user to choose a function, the user has only to type the letter of the function he wishes to perform. Upon doing this, PSS will provide the appropriate responses. Error messages for such things as invalid data or entering a letter which is not one of the menu functions shown are printed in an area at the bottom of the screen reserved for error messages. The cursor then returns to the place on the screen where the error was made and the user is given a chance to try again. PSS will continue to do this until a valid response is entered.

At any level of PSS, a quit command (entered by typing a "Q" as the response to any menu) will allow the user to stop the current operation and return to the previous menu. Users must back their way out of the system one menu at a time. They also have the opportunity to back up to any previous menu and then do something different from what they originally did.

Whenever a PSS function involving an individual student is invoked, the CRT screen is always divided into two areas (in addition to the "error footer" at the bottom of the screen), an upper area called the STUDENT HEADER and a center area for whatever screen display is appropriate to the function that was invoked.

The format for the STUDENT HEADER is:

RANK	LAST NAME, FIRST NAME MI.	FULL SSAN
SCHOOL	PROGRAM	OUTSTANDING SUSPENSE FLAG

EXAMPLE:

2Lt	Dimwitter, Jonathan I.	999-55-1234
Podunk University	POCDY	*

NOTE: 1. The suspense flag will be flashing if there are any outstanding suspenses for the student, otherwise it is blank.
2. These headers are displayed as write protected fields and cannot be directly changed by the user.

properly entered, by using the appropriate password. Next, the CI Authorization/Access Check program (CIAAC) must be invoked. Once CIAAC has displayed SCREEN 0, a CADIS password and a CI user ID must be entered. This is done by entering the correct information in the right fields of SCREEN 0.

The format for SCREEN 0 is:

```
ENTER PASSWORD [          ] AND
    CI USER ID [    ]
```

The CADIS password will be used by PSS latter to access CADIS as necessary. The CI user ID is used to do two things. The first is to logically partition the CI data in CADIS into two parts, that which a given PM is responsible for (and can directly access), and that which he is not responsible for (and cannot directly access), i.e. select the proper users view. This first part is done by the DBMS. Note: The format of the CI user ID is a four character combination which begins with CI and ends with a two letter code uniquely identifying a given program manager, e.g. CIRA.

The second thing the CI user ID does is to allow PSS to automatically direct the user to the appropriate set of user functions (i.e. CIR/M or CIV functions). These two tasks would be the responsibility of a module called "Authorization/Access Checking". This module has not been defined in greater detail at this time, since it will be heavily dependent upon the security capabilities and details of whichever DBMS is chosen to be the basis of CADIS.

Assuming the user entered an authorized CI users ID and a correct CADIS password, the next thing he will see is SCREEN 1, the PSS entry screen.

The format for SCREEN 1 is:

```
    SSAN: [     ]        PM: [  ]
SUSPENSE: [ ]          TYPE: [ ]
    DATE: [        ]
  CREATE: [ ]
    QUIT: [ ]
```

When SCREEN 1 is output to the CRT, the last two characters of the CI users ID entered in SCREEN 0 and today's date are displayed in the appropriate fields, and all other fields are blank. At this point the user has the option of changing the program managers code (the last 2 characters of the CI users ID), of viewing all suspenses, creating a student header (input minimal data to enter a new CI student in CADIS), processing an individual student who is already in the system, or quitting (i.e. return to SCREEN 0).

NOTE: Assuming the user needs to do anything involving another PM's area of responsibility (view), then he must enter the last two characters of the other PM's CI users ID into the PM field. Changing the PM field is allowed to permit an authorized program manager to examine/change student files of another PM who is not able to perform these tasks for some reason, e.g., the other PM is on leave.

If the user desires to see a list of all his suspenses, he has several ways of specifying what he would like to see. If he enters a "Y" in SUSPENSES without entering anything else on the screen, he will see a list of all suspenses up to and including today's date. If a date other than todays had been entered in DATE then the list of suspenses retrieved from CADIS would be all those suspenses up to and including the specified date. If a value had been entered in TYPE then only suspenses of that particular type, up to and including whatever date was in DATE would be listed. (Default value for TYPE is blank, resulting in all types of suspenses being retrieved, subject only to DATE constraints.) DATE is a seven character field for day, month, and year, e.g. 28 NOV 82.

In the event that the user would like to enter data concerning a new CI student into CADIS, the first thing he must do is to create a new student header. (As a by product of entering the appropriate data into the new student header, the appropriate minimal data is obtained to create all necessary relations for the new student in CADIS.) All that is required to initiate the process is to enter a "Y" in CREATE. This causes SCREEN 1C, the create new student screen, to be displayed.

The format for SCREEN 1C is:

```
ENTER DATA ON NEW STUDENT IN APPROPRIATE FIELDS

RANK [     ]  NAME (LAST, FIRST MI) [                                   ]
FULL SSAN [          ]
SCHOOL [                              ]     PROGRAM [     ]

ENTER A "Q" IN SSAN TO QUIT WITHOUT FINISHING STUDENT HEADER
```

If the user would like to process an individual student, he must enter the last four digits of the student's SSAN into the SSAN field of SCREEN 1.

Finally, if the user desires to quit, i.e. leave SCREEN 1 and return to SCREEN 0, then he enters a "Q" in QUIT.

Assuming that on SCREEN 1 the user had elected to view a list of all suspenses, the next screen he would see is screen 1A, the general suspense list screen.

The format of SCREEN 1A is:

```
AUTOMATIC PAGING (Y/N): [ ]

LAST NAME, FIRST NAME MI.     FULL SSN    DATE   TYPE   ACTION
LAST NAME, FIRST NAME MI.     FULL SSN    DATE   TYPE   ACTION
```

NOTE: If a "Y" is entered in the auto paging field, the listing automatically stops after one screen of data has been displayed. To start the listing up again and get the next screen of data, simply hit "RETURN". Entering an "N" in the auto paging field lets the listing be printed out continuously from start to finish.

In the event that the user had decided to process an individual student on SCREEN 1, it is possible that more than one student possesses the same last four digits in their SSAN as well as also being the responsibility of a given user (PM). If this does happen, then the user will see SCREEN 1B, the duplicate student screen.

The format of SCREEN 1B is:

```
ENTER FULL SSN FOR DESIRED STUDENT [          ]

LAST NAME, FIRST NAME MI.     FULL SSAN
LAST NAME, FIRST NAME MI.     FULL SSAN

ENTER A "Q" FOR SSAN TO QUIT WITHOUT SELECTING A STUDENT
```

By selecting the full SSAN of the student desired and entering it on SCREEN 1B, the user and PSS is assured of processing only the a single correct student.

Assuming the decision to process an individual student was made at SCREEN 1, it is at this point (i.e., a single student has been identified) that PSS builds the STUDENT HEADER (see the description of a STUDENT HEADER above). As mentioned previously, for all subsequent screen displays involving this student, his STUDENT HEADER will be the write protected top portion of the CRT screen.

For processing individual students PSS has four functions which can be invoked by the user. SCREEN 2 shows these four individual student functions as well as the common function QUIT.

```
The format for SCREEN 2 is:

(STUDENT HEADER)

          A. BIOGRAPHICAL DATA
          B. EDUCATIONAL PLAN
          C. MEMO FOR THE RECORD
          D. SUSPENSES
          Q. QUIT

ENTER YOUR SELECTION: [ ]
```

NOTE: The commands defined earlier in the functional description, e.g. "+", "-", apply to all the individual student functions and screens that follow.

If the user enters an "A" on SCREEN 2, the next screen that he will see is SCREEN 3, the bio data screen.

The format for SCREEN 3 is:

(STUDENT HEADER)

```
LINE CMD
  1  [ ] GRADUATE GPA [     ]   BEGIN DATE    [      ]  ED COL:           [     ]
  2  [ ] PROGRAM MGR  [     ]   GRAD  DATE    [      ]  DEGREE CODE       [      ]
  3  [ ] ACADEMIC STATUS [ ]    CLASS CODE    [      ]  DEGREE GRANTED [         ]
  4  [ ] ADVISOR [          ]   TRS-IN-DATE   [      ]  DEPT-DATE [       ]
  5  [ ] MANNING CODE   [  ]    TRS-OUT-DATE [       ]  AVAIL-DATE [       ]
  6  [ ] LAST MAJCOM    [    ]  DEPARTURE REASON   [                     ]
  7  [ ] OUTPUT MAJCOM  [    ]  STATUS     [                               ]
  8  [ ] PAFSC   [       ]      DUTY TITLE [                                ]
  9  [ ] THESIS PAY [ ]         DUTY PHONE [    ] [       ]
 10  [ ] U-GRAD GPA [     ]  U-GRAD SCHL [                                 ]
 11  [ ] PRIOR DEGREE   [     ] PRIOR AFIT [   ]
 12  [ ] GRE    VERBAL [    ]   QUANT [    ]              TOTAL   [    ]
 13  [ ] GMAT   TOTAL   [    ]
 14  [ ] DOR [        ]         TMST   [       ]          AERO RATING [ ]
 15  [ ] DOB [        ]  MARITAL STATUS [ ]  ETHNIC GROUP [ ]   SEX [ ]
 16  [ ] LOCAL ADDRESS [                                                   ]
 17  [ ] HOME PHONE [    ] [       ]   STATE-OF-LEGAL-RESIDENCE [  ]
```

NOTE: ALL GPA'S SHOWN ON SCREEN 4 HAVE BEEN CONVERTED TO A 4.0 SCALE

Entering a "B" on SCREEN 2 results in SCREEN 4, the ed plan, being displayed. When SCREEN 4 is first displayed on the CRT, it will show the current term. Paging back, i.e., entering a "-" command in the CMD field of LINE 1, will show the the user the previously completed term(s). Paging forward from the original SCREEN 4 displayed, i.e., entering a "+" command in the CMD field of LINE 1, will show the student's planned schedule.

The format for SCREEN 4 is:

```
(STUDENT HEADER)

LINE CMD
  1  [ ]  PROJECTED GRADUATION DATE: [       ]
  2  [ ]  DATE OF LATEST APPROVED ED PLAN: [       ]
          TERM #     TERM START DATE       TERM STOP DATE     TERM TYPE
  3  [ ]  [  ]          [       ]             [       ]          [ ]
          COURSE   COURSE     COURSE       COURSE         CREDIT   GRADE
           ID       DEPT       NUM         TITLE            HRS
  4  [ ]  [   ]    [    ]    [      ]  [               ]  [  ]      [  ]
  5  [ ]  [ · ]    [    ]    [      ]  [               ]  [  ]      [  ]
  .
  .
  n  [ ]  [   ]    [    ]    [      ]  [               ]  [  ]      [  ]

                                           TOTAL TERM HRS [  ] GPA [   ]
```

If a "C" had been entered on SCREEN 2, then SCREEN 5, the memo for record screen (MFR), will be displayed.

The format for SCREEN 5 is:

```
(STUDENT HEADER)

LINE CMD    DATE                MEMO
  1  [ ]  [       ]  [                                                    ]
                     [                                                    ]
  2  [ ]  [       ]  [                                                    ]
                     [                                                    ]
  3  [ ]  [       ]  [                                                    ]
                     [                                                    ]
  .
  .
```

NOTE: MFR'S are considered as one record per two lines, even though they are displayed on two lines. When the end of the first block is reached, the terminal will automatically jump to the next block.

Assuming the user had entered a "D" on SCREEN 2 then SCREEN 6, the individual student's suspense screen, is displayed.

The format for SCREEN 6 is:

```
(STUDENT HEADER)

LINE CMD    DATE      TYPE                ACTION
  1  [ ]  [         ] [    ]    [                                                  ]
  2  [ ]  [         ] [    ]    [                                                  ]
  3  [ ]  [         ] [    ]    [                                                  ]
  .
  .
  n [ ]   [         ] [    ]    [                                                  ]
```

MODULE DESCRIPTIONS FOR THE AFIT/CIR/M PERSONNEL SUBSYSTEM SOFTWARE (PSS)

Further information on any of the menus or displays can be found in the Functional Description (FD) of PSS, APPENDIX IIIC.

MODULE 1

```
*******************************************************************
*                                                                 *
*NAME:   AUTHORIZATION/ACCESS CHECKING                            *
*                                                                 *
*FUNCTION: Displays SCREEN 0; Checks for proper database password *
*        and CI users ID; Routes users to proper subsystem (PSS/FSS)*
*INPUT:    DATABASE PASSWORD AND CI USER ID                       *
*OUTPUT:   UID (CI users ID), PW (Database Password)              *
*******************************************************************
  PROCEDURE:
    TO BE DEVELOPED (TBD)
```

MODULE 2

```
*******************************************************************
*                                                                 *
*NAME:  FINANCIAL  SUBSYSTEM (FSS)                                *
*                                                                 *
*FUNCTION:  See APPENDIX IIIG                                     *
*                                                                 *
*INPUT:                                                           *
*OUTPUT:                                                          *
*******************************************************************
  PROCEDURE:
```

MODULE 3

```
********************************************************************
*                                                                  *
*NAME:  Personnel Subsystem (PSS)                                  *
*                                                                  *
*FUNCTION:  Driver for entire Personnel Subsystem Software;        *
*           Displays SCREEN 1                                      *
*INPUT:  UID, PW                                                   *
*OUTPUT: UID, PW, SHORTSSAN (last 4 digits of SSAN)                *
********************************************************************
```

```
PROCEDURE:
       Begin
         HALT = false
         While not HALT do
            Begin
                  Print SCREEN 1
                  SCREEN = "1"
                  Repeat
                     Call Get_Input (SCREEN,INPUT,DONE)
                     Call Validate_Input (INPUT,AMV,VALIDI   T)
                     If not VALIDINPUT then
                        Call Flag_Errors (AMV)
                     Endif
                  Until DONE & VALIDINPUT
                  Case (nonblank & validated) INPUT of fiel  ,
                    1. PM not = (last 2 char. of UID): Call  hange_PM
                       (UID)
                    2. SSAN & PM: begin
                                  Call Get_Full_SSAN(SHORT  AN,
                                  UID,PW,FULLSSANLIST,SSA  UNT,
                                  SSAN)
                                  If SSANCOUNT > 1 then
                                    Call Process_Duplicate
                                    Students (UID,PW,FULLS  NLIST,
                                    SSANCOUNT,FULLSSAN)
                                  Endif
                                  If SSANCOUNT >= 1 then
                                     Call Build_Student_He  ler(
                                     UI  ,PW,FULLSSAN,SHR);  SHR
                                     means "Student Header  ecord"
                                     Call Process_Student(  D,PW,
                                     SHR)
                                  Endif
                                 End(2)
                    3. Suspense & Date & Type: Call Proces  eneral_
                       Suspenses(UID,PW,DATE,TYPE)
                    4. Create = "Y": Begin
                                     Call Create_New_Stud  (PW,UID,
                                     SHR)
                                     Call Build_Student_He  er(UID,
                                     PW,FULLSSAN,SHR)
                                     CALL Process_Student(  D,PW,SH
                                    End(4)
                    5. Quit= "Q": HALT = true
                  Endcase
           End(while)
       End (main)
```

```
MODULE 4
*****************************************************************************
*                                                                           *
*NAME: Change_PM                                                            *
*                                                                           *
*FUNCTION: To change CI users ID w/o backing up to Module 1                 *
*                                                                           *
*INPUT:  UID                                                                *
*OUTPUT: UID                                                                *
*****************************************************************************
 PROCEDURE: TBD


MODULE 5
*****************************************************************************
*                                                                           *
*NAME: Create_New_Student                                                   *
*                                                                           *
*FUNCTION: To enter a new CI student into the database; Create              *
*          and Display STUDENT HEADER                                       *
*INPUT:  PW,UID                                                             *
*OUTPUT: SHR (Student Header Record)                                        *
*****************************************************************************
 PROCEDURE:
   Begin
     Print out SCREEN IC
     Repeat
       Call Get_Input(SCREEN,INPUT,DONE)
       Call Validate_Input(INPUT,AMV,VALIDINPUT)
       If not VALIDINPUT then
         call Flag_Errors (AMV); note AMV="appropriate module variables"
       Endif
       If SSAN= "Q" then
         DONE= true; note, Validate_Input accepts "Q" as a valid SSAN
       Endif
     Until DONE & VALIDINPUT
     If SSAN not = "Q" then
       Send data to database
       Store data in SHR; SHR="Student Header Record"
     Endif
   End (main)
```

```
MODULE 6
**************************************************************** *******
*                                                                      *
*NAME: Build_Student_Header                                            *
*                                                                      *
*FUNCTION: Build and Display the STUDENT HEADER                        *
*                                                                      *
*INPUT: UID,PW,FULLSSAN                                                *
*OUTPUT: SHR (Student Header Record)                                   *
**************************************************************** *******
 PROCEDURE:
   Begin
     If the SHR for this student doesn't already exist then
       Retrieve data from database
       Store data in SHR
     Endif
     Print out the STUDENT HEADER
   End (main)
```

```
MODULE 7
**************************************************************** *******
*                                                                      *
*NAME: Process_Student                                                 *
*                                                                      *
*FUNCTION: Driver for individual student management modules;          *
*          Display SCREEN 2                                            *
*INPUT: UID,PW,SHR (Student Header Record)                             *
*OUTPUT: UID,PW,SHR                                                    *
**************************************************************** *******
 PROCEDURE:
   Begin
     Print out SCREEN 2
     SCREEN ="2"
     Repeat
       Call Get_Input (SCREEN,INPUT,DONE)
       Call Validate_Input (INPUT,AMV,VALIDINPUT)
       If not VALIDINPUT then
         Call Flag_Errors (AMV); note AMV="appropriate module variables"
       Endif
     Until DONE & VALIDINPUT
     Case Input of
       1. "A":Call Process_Bio_Data(UID,PW,FULLSSAN)
       2. "B":Call Process_Ed_Plan(UID,PW,FULLSAN)
       3. "C":Call Process_Memo_For_Record(UID,PW,FULLSSAN)
       4. "D":Call Process_Individual_Suspense(UID,PW,FULLSSAN)
       5. "Q":Null; no action, fall to end of procedure & return
                    to module 3
     Endcase
  End (main)
```

```
MODULE 8
********************************************************************
*                                                                  *
*NAME: Process_General_Suspenses                                   *
*                                                                  *
*FUNCTION: Display SCREEN 1A; Retrieve and Display ALL the         *
*          outstanding suspenses of a particular type (TYPE to be  *
*          determined) for ALL students                            *
*INPUT: UID,PW,DATE,TYPE                                           *
*OUTPUT: None                                                      *
********************************************************************
 PROCEDURE:
   Begin
     Print out SCREEN 1A
     SCREEN = "1A"
     Repeat
       Call Get_Input(SCREEN,INPUT,DONE)
       Call Validate_Input(INPUT,AMV,VALIDINPUT)
       If not VALIDINPUT then
         call Flag_Errors(AMV); AMV="appropriate module variables"
       Endif
     Until DONE & VALIDINPUT
     If INPUT = "Y" then
       Repeat
         If (Screen isn't full) then
           Retrieve data from the database
         Else
           call Get_Char(INPUT)
       Until (no more data)
     Else
       Repeat
         Retrieve data from the database
       Until (no more data)
  End (main)


MODULE 9
********************************************************************
*                                                                  *
*NAME: Get_Full_SSAN                                               *
*                                                                  *
*FUNCTION: Retrieve all SSAN's from the database whose last        *
*          digits = SHORTSSAN & where the SSAN is in the UID's     *
*          view                                                    *
*INPUT: SHORTSSAN,PW,UID                                           *
*OUTPUT: FULLSSANLIST,SSAN,SSANCOUNT                               *
********************************************************************
 PROCEDURE:
   Begin
     SSANCOUNT = 0
     Repeat
       Retrieve data from the database
       Put data into FULLSSANLIST
       Increment SSANCOUNT
     Until (no more data)
     If SSANCOUNT < 1 then
       Call Flag_Errors(AMV); AMV="appropriate module variables
     Endif
   End (main)


MODULE 10
*******************************************************************
*                                                                 *
*NAME:  Process_Duplicate_Students                                *
*                                                                 *
*FUNCTION: Display SCREEN 1B; Retrieve & Display data on all      *
*          students who have the same last 4 digits as those      *
*          digits entered into SSAN on SCREEN 1; Get the full     *
*          SSAN of the single student the user desires to process *
*INPUT: UID,PW,FULLSSANLIST,SSANCOUNT                             *
*OUTPUT: FULLSSAN, SSANCOUNT                                      *
*******************************************************************
 PROCEDURE:
   Begin
     Print SCREEN 1B
     For I = 1 to SSANCOUNT do
       Retrieve data for student whose SSAN = Ith SSAN in
       FULLSSANLIST
       Print out data
     Endfor
     SCREEN = "1B"
     Repeat
       Call Get_Input(SCREEN,INPUT,DONE)
       Call Validate_Input(INPUT,AMV,VALIDINPUT)
       If not VALIDINPUT then
         Call Flag_Errors(AMV)
       Endif
       If INPUT = "Q" then
         DONE = true; Validate_Input accepts "Q" as valid
       Endif
     Until DONE & VALIDINPUT
   End (main)
```

```
MODULE 11
********************************************************************
*                                                                  *
*NAME:  Process_Bio_Data                                           *
*                                                                  *
*FUNCTION: Display SCREEN 3; Retrieve & Display Biodata; Drive     *
*          Module 15                                               *
*INPUT:  UID,PW,FULLSSAN                                           *
*OUTPUT: CMD,UID,PW,FULLSSAN (CMD is the line command variable)    *
********************************************************************
 PROCEDURE:
   Begin
     HALT= false
     While not HALT do
       Repeat
         retrieve data from the database
       Until (no more data)
       Print out SCREEN 3
       SCREEN = "3"
       Repeat
         Call Get_Input(SCREEN,INPUT,DONE)
         Call Validate_Input(INPUT,AMV,VALIDINPUT)
         If not VALIDINPUT then
           Call Flag_Errors(AMV); AMV="appropriate module variables"
         Endif
       Until DONE & VALIDINPUT
; NOTE: If any line commands were entered on SCREEN 3 then Get_Input &
        Validate_Input will have assigned a valid value to each line
        command (CMD)
       Repeat
         If (CMD not = " ") & (CMD not = "Q") then
           Call Process_Line_Command(UID,PW,FULLSSAN,CMD,SCREEN)
         Endif
       Until (All line commands have been processed) or (CMD = "Q")
       If CMD = "Q" then
         HALT = true
       Endif
    Endwhile
  End (main)
```

MODULE 12

```
***************************************************************************
*                                                                         *
*NAME:  Process_Ed_Plan                                                   *
*                                                                         *
*FUNCTION: Display SCREEN 4; Retrieve & Display Ed Plan data;             *
*           Drive module 15                                               *
*INPUT:   UID,PW,FULLSSAN                                                 *
*OUTPUT:  CMD, UID, PW, FULLSSAN (CMD is the variable for line            *
*         command values)                                                 *
***************************************************************************
```

PROCEDURE:

Same as module 11 except replace "SCREEN 3" references with "SCREEN 4" references where appropriate

MODULE 13

```
***************************************************************************
*                                                                         *
*NAME: Process_Memo_For_Record                                            *
*                                                                         *
*FUNCTION: Display SCREEN 5; Retrieve & Display MFR data                  *
*                                                                         *
*INPUT:   UID,PW,FULLSSAN                                                 *
*OUTPUT:  CMD,UID,PW,FULLSSAN (CMD is the variable for line               *
*         command values)                                                 *
***************************************************************************
```

PROCEDURE:

Same as module 11 except replace "SCREEN 3" references with "SCREEN 5" where appropriate

MODULE 14

```
***************************************************************************
*                                                                         *
*NAME:  Process_Individual_Suspense                                       *
*                                                                         *
*FUNCTION: Display SCREEN 6; Retrieve & Display suspense data for         *
*           an individual student; Drive module 15                        *
*INPUT:  UID,PW,FULLSSAN                                                  *
*OUTPUT: CMD,UID,PW,FULLSSAN (CMD is the variable for line command        *
*        values)                                                          *
***************************************************************************
```

PROCEDURE:

Same as module 11 except change "SCREEN 3" references to "SCREEN 6" where appropriate

```
MODULE 15
*****************************************************************
*                                                               *
*NAME:  Process_Line_Command                                    *
*                                                               *
*FUNCTION: Drive the line command processor modules             *
*                                                               *
*INPUT:  UID,PW,FULLSSAN,CMD,SCREEN (see module 14 for CMD explain)*
*OUTPUT: UID,PW,FULLSSAN,SCREEN                                 *
*****************************************************************
 PROCEDURE:
   Begin
     Case CMD of
       1. "+": Call Page_Forward(UID,PW,FULLSSAN,SCREEN)
       2. "-": Call Page_Back(UID,PW,FULLSSAN,SCREEN)
       3. "C": Call Change_Record(UID,PW,FULLSSAN)
       4. "D": Call Delete_Record(UID,PW,FULLSSAN)
       5. "S": Call Select_Record(UID,PW,FULLSSAN)
       6. "N": Call Create_New_Record(UID,PW,FULLSSAN,SCREEN
     Endcase
   End (main)
```

```
MODULE 16
*****************************************************************
*                                                               *
*NAME:  Create_New_Record                                       *
*                                                               *
*FUNCTION: Display appropriate basic screen with no data;       *
*          Receive new data from keyboard; Transmit data to database*
*INPUT:  UID,PW,FULLSSAN,SCREEN                                 *
*OUTPUT: None                                                   *
*****************************************************************
 PROCEDURE:
   Begin
     Print appropriate screen
     Repeat
       Call Get_Input(SCREEN,INPUT,DONE)
       Call Validate_Input (INPUT,AMV,VALIDINPUT); NOTE: LINE CMD's
       If not VALIDINPUT then                      ; except "Q" not
         Call Flag_Errors(AMV)                     ; accepted
       Endif
       If (VALIDINPUT) & (INPUT not = "Q") then
         Transmit data to database as appropriate
       Endif
       If INPUT = "Q" then
         DONE = true; Validate_Input accepts "Q" as a valid input
       Endif
     Until DONE & VALIDINPUT
   End (main)
```

```
MODULE 17 & 18
*****************************************************************************
*                                                                           *
*NAME: Page_Forward/Back                                                    *
*                                                                           *
*FUNCTION: Show next/previous page of data of screen currently              *
*          being displayed                                                  *
*INPUT:   UID,PW,FULLSSAN,SCREEN                                            *
*OUTPUT:  None                                                              *
*****************************************************************************
 PROCEDURE:
   Begin
     Get Next/Previous page of data
     Print appropriate screen
   End (main)
```

```
MODULE 19,20,21
*****************************************************************************
*                                                                           *
*NAME: Change/Delete/Select_Record                                          *
*                                                                           *
*FUNCTION: Change/Delete/Select appropriate record on the screen            *
*          and in the database; Update screen display as necessary          *
*INPUT: UID,PW,FULLSSAN                                                     *
*OUTPUT: None                                                               *
*****************************************************************************
 PROCEDURE:
   TBD
```

```
MODULE 22
*****************************************************************************
*                                                                           *
*NAME:  Get_Input                                                           *
*                                                                           *
*FUNCTION: This is a submodule which is different in specifi                *
*          details for each module which uses it, but is ver                *
*          similiar in function.  It gets a line of input from a            *
*          screen & assigns the input to the appropriate variables          *
*          for a given line of that screen                                  *
*INPUT: SCREEN,INPUT,DONE (SCREEN indicates which screen display            *
*        is being used, DONE is a boolean variable set to tru by            *
*        Get_Input when all the variables associated with a given           *
*       screen have been assigned values (all input from the screen         *
*        has been read in), INPUT is a temporary variable used to           *
*        hold input from the screen).                                       *
*OUTPUT: NONE                                                               *
*****************************************************************************
 PROCEDURE:
   TBD
```

MODULE 23

```
*************************************************************************
*                                                                       *
*NAME:  Validate_Input                                                  *
*                                                                       *
*FUNCTION: A submodule which is different in detail for each            *
*          module that uses it, but is similar in function for          *
*          all of them.  This module validates that the input from      *
*          a given field for a given screen is correct in format,       *
*          range, type, etc. as desired.  This could be done by         *
*          referencing the data dictionary and other appropriate        *
*          sources.                                                     *
*INPUT: INPUT,VALIDINPUT,AMV (VALIDINPUT is a boolean variable          *
*       which is set to true if the input for a given AMV is valid      *
*       and which is set to false otherwise. AMV refers to approp-      *
*       riate module variable.  INPUT is a temporary var. used for      *
*       reading in data.)                                               *
*OUTPUT: VALIDINPUT                                                     *
*************************************************************************
 PROCEDURE:
   TBD
```

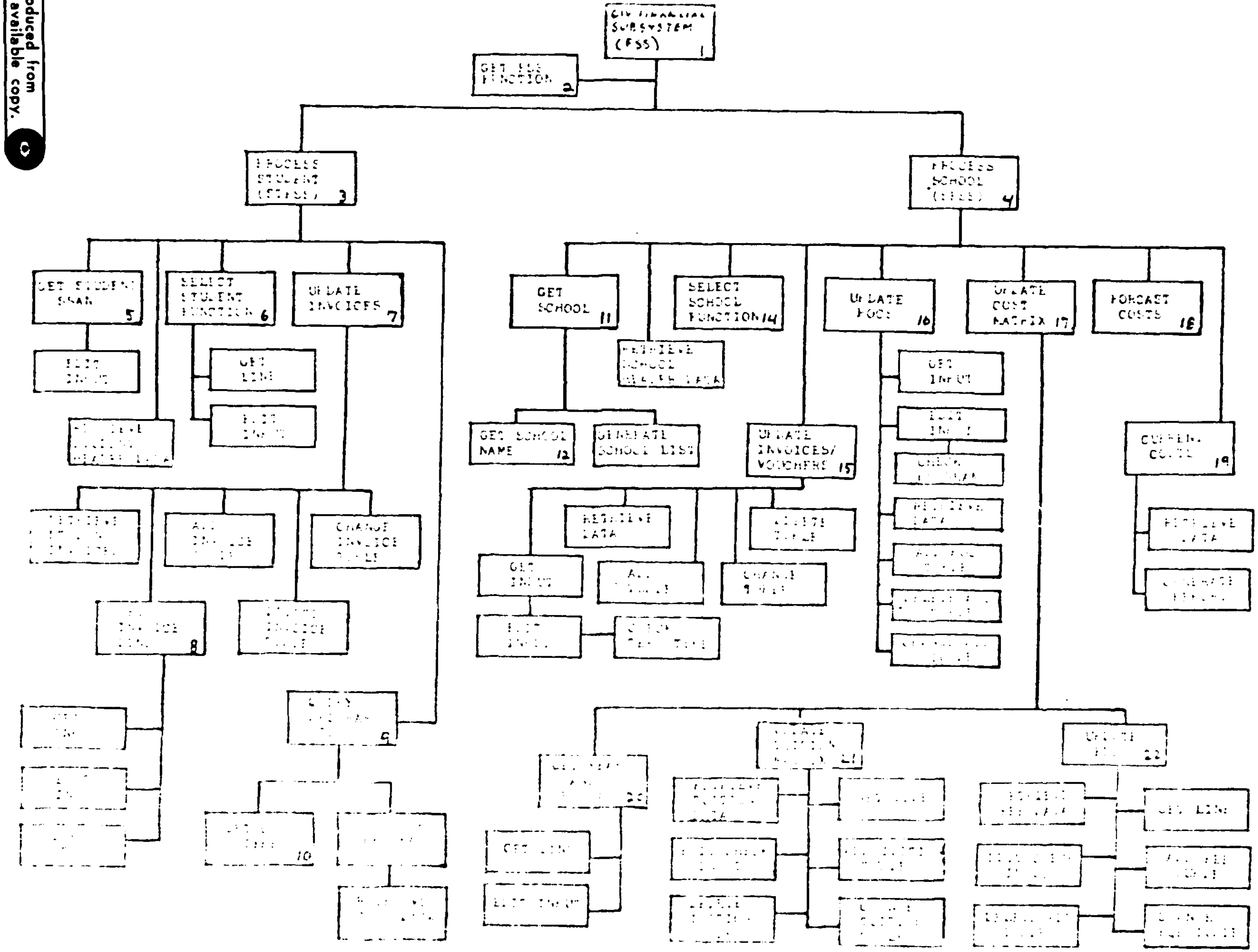
MODULE 24

```
*************************************************************************
*                                                                       *
*NAME:  Flag_Errors                                                     *
*                                                                       *
*FUNCTION: This submodule is different in detail for every module       *
*          which uses it, but is very similiar in function.  This       *
*          module would be designed to print out appropriate error      *
*          messages in the ERROR FOOTER at the bottom of the page       *
*          and to then return the cursor to the point of the error.     *
*INPUT:  AMV (appropriate module variables)                             *
*OUTPUT: None                                                           *
*************************************************************************
 PROCEDURE:
   TBD
```

```
MODULE 25
************************************************************************
*                                                                      *
*NAME: Get_Char                                                        *
*                                                                      *
*FUNCTION: Gets one character from the terminal.  Maintains program*
*          control, and waits doing nothing until a char. is input *
*          Does nothing with the char. & then returns to calling   *
*          program.                                                    *
*INPUT:  INPUT                                                         *
*OUTPUT: None                                                          *
************************************************************************
 PROCEDURE:
   TBD
```

OTHER AREAS WHICH WILL NEED TO BE DEVELOPED DUE TO INSUFFICIENT DATA AT THIS TIME:

Procedures to get next/previous page of data Procedures to retrieve/transmit data to/from the database



Reproduced from best available copy.

Appendix III E

A FUNCTIONAL DESCRIPTION OF THE

AFIT/CIV FINANCIAL SUBSYSTEM (FSS) SOFTWARE

The following is a description of the AFIT/CIV Financial Subsystem (FSS) from the users standpoint. For each menu which requests the user to choose a function, the user has only to type the letter of the function he wishes to perform. Upon doing this, the program will provide the appropriate responses. Error messages for such things as invalid input or entering a letter which is not displayed are printed at the bottom of each screen and the user is given a chance to try again. The program will continue to do this until a valid response is entered.

At every level, the quit or 'Q' will allow the user to stop the the current operation and return to the previous menu. Users must back their way out of the system one menu at a time. They also have the opportunity to go back and do something different.

The security and entry procedures for the FSS are identical to those used in the Personnel Sub System (PSS). Further information about this procedure can be obtained by referring to appendix IIIC.

The two displays STUDENT HEADER (D1) and SCHOOL HEADER (D2), are displayed depending on whether the user is performing a student or school operation. These headers are displayed as protected fields and cannot be directly changed by the user.

In the update programs the display is initialized with the cursor in the function column (FUN) of the first line. The user has only to move the cursor up or down to the line that is to be updated ,enter the proper function (A, Add; C, Change; or D, Delete) and move the cursor to the begining of the field(s) he wishes to modify. Once finished, the user transmits the line and the program will edit each field according to predefined criteria (e.g. SSAN is all numeric). The first item checked is the line number. If this has been changed or is unrecognizable, the program will harshly notify the user and restore the line to its original format. If the user is updating line 3 of a ten line display, for instance, and changes the line number to 6; the update program will think he is operating on line 6 and make erroneous changes. THE USER MUST NOT CHANGE THE LINE NUMBERS or the results cannot be predicted.

Errors in the update programs are handled just like any other error and are displayed at the bottom of each page. The line is scanned from left to right and one message is displayed for each error. When the first error is detected, the program stops and prints the message and terminates scanning. Therefore, if a user has three errors in a line he will get three separate error messages and have to make three changes. The software does not search for all errors, at one time, but simply notifies the user of the first one it finds.

Upon entering the system, through the proper security procedures described earlier, the user will first be presented with the following menu:

```
MENU V1: CHOOSE THE FUNCTION TO BE PERFORMED:  [ ]

         A:  STUDENT PAYMENTS
         B:  SCHOOL PAYMENTS
         Q:  QUIT
```

At this point the user must choose whether he will operate on student information, school information or quit altogether.

Should the user desire choose a student operation, he will be asked to enter the full SSAN of the student using menu V2.

```
    MENU V2:  PROCESS A STUDENT

         ENTER STUDENTS FULL SSAN [          ]
```

Once a student has been retrieved from the database, the system will display the student header format (D1) for each operation dealing with that student. The user may not change this header directly.

```
    D1:  STUDENT HEADER
SSAN        RANK    NAME                      MAF         FUND_TYPE
ADDRESS                     SCHOOL            START_DATE  GRAD_DATE
                           PM         TERM_TYPE
                 STATE_OF_LEGAL_RES           PHONE
-------------------------------------------------------------------
EXAMPLE:
999-55-1234  2LT   DIMWITTER, JONATHAN I.     POCAY       FUNDED LEVEL
1234 HOLE BLVD             PODUNK UNIVERSITY  3 JAN 51    22 DEC 99
ROOM 125                   CIRW          SEM
MOOSEJAW, WI  00001        ND                (111)-123-3345
```

After the student header is displayed, the user can next specify whether he wants to work on the invoices paid for that student or simply look at a list of the costs/expenses incurred on behalf of that student, to date. Menu V2.1 will also let the user go back and identify another student to process.

```
MENU V2.1:  SELECT THE OPTION TO PROCESS

STUDENT HEADER (D1)

SELECT THE DESIRED FUNCTION TO BE PERFORMED  [ ]
     A:  UPDATE/EXAMINE INVOICES
     B:  EXAMINE STUDENT COSTS
     C:  EXAMINE ANOTHER STUDENT
     Q:  QUIT
```

Should the user desire to examine or update the invoices for the student, the data will be displayed in the format of menu/display V2.1.1. This is the basic format for all of the update programs, with the variable data being placed within a set of brackets ([ ]) and error messages printed at the bottom of the display. The brackets restrict the size of the input field and give the user an idea as to how large it is. If for some reason there are too many lines to display on a single screen, the user will be able to "page" through them by entering a "+" or "-" under the function column. This will move the display forward (+) or backward (-) one screenfull.

```
MENU V2.1.1:  UPDATE/EXAMINE STUDENT INVOICES

STUDENT HEADER (D1)

           VOUCHER_NUMBER    VOUCHER_AMOUNT    PMT_CODE     INCLUSIVE_DATES
           DATE_POSTED          AMOUNT_REQUESTED  TERM_TYPE        REMARKS
LINE FUN
-------------------------------------------------------------------------------
1    [ ]   [              ]  $[            ]     [    ]    [        .        ]
           [       ]            $[            ]     [   ]  [                 ]
2    [ ]   [              ]  $[            ]     [    ]    [        .        ]
           [       ]            $[            ]     [   ]  [                 ]
:
:
:

EXAMPLE
           VOUCHER_NUMBER    VOUCHER_AMOUNT    PMT_CODE     INCLUSIVE_DATES
           DATE_POSTED          AMOUNT_REQUESTED  TERM_TYPE        REMARKS
LINE FUN
-------------------------------------------------------------------------------
1    [ ]   [123456789AB]    $[123456.00]         [FEES]    [01JAN80-12MAR81]
           [15JAN80]           $[987654.99]         [SEM]  [ASKING TOO MUCH ]
```

Should the user desire to examine the various costs accrued by the student, he can instead select option B from menu V2.1 and get the following display. This does not produce an updateable displ ; menu V2.1.2 displays only the student costs. The user simply se cts the cost, the options are erased and the retrieved data is displayed

```
    MENU V2.1.2:  EXAMINE STUDENT COSTS

    STUDENT HEADER
    SELECT THE DESIRED STUDENT COSTS
         A:  TUITION
         B:  LABRATORY
         C:  BOOKS
         :
         :
         Q:  QUIT
```

```
D2:  STUDENT COSTS

STUDENT HEADER
COST_TYPE   SCCA       FY82       FY81       FY80        FY79          TAL
-------------------------------------------------------------------------
TUITION    [      ]  [      ]  [      ]  [       ]  [       ]  [         ]
LABRATORY  [      ]  [      ]  [      ]  [       ]  [       ]  [         ]
BOOKS      [      ]  [      ]  [      ]  [       ]  [       ]  [         ]
 :
 :
 :
```

If the user had chosen option B from menu V1, he would be · erating on information concerning a school instead of a student. After ntering option B, the a menu will appear (menu V3) which will allow the ser to select the particular school he wishes to see.

At this point, the user may enter the exact name of the sch l (e.g. AUBURN UNIVERSITY) or simply enter the two letter state abbrevi ion and a list of schools will be generated. The user may then choose school from the list and continue. If an incorrect name is entered, th program will attempt to find it then return with an error message, menu V3 and let him try again.

Menu V3 will allow the user to select the name of the s ool he desires.

```
MENU V3:  SELECT THE SCHOOL

ENTER ONE OF THE FOLLOWING:

          A:  SCHOOL NAME [                                  ]
          B:  SCHOOL STATE [   ]
          Q:  QUIT [ ]
```

If the user requests a list of schools in a particular sta , menu V3.1 will appear and allow him to select a school from the li . This name will be automatically transferred to the routine which will etrieve the school data.

```
MENU V3.1:  GENERATE SCHOOL LIST

SELECT THE DESIRED SCHOOL

          LINE  FUN  SCHOOL NAME
            1   [ ]  [                                  ]
            2   [ ]  [                                  ]
            3   [ ]  [                                  ]
            :
            :
            N   [ ]  [QUIT]
```

Once a school has been selected, the system will auto tically display a school header which contains certain information per nent to the school. This header (D3) cannot be changed directly by the r and remains on the screen throughout the operations for that school.

DISPLAY D3: SCHOOL HEADER

```
SCHOOL_NAME                    ESA_NUMBER
ADDRESS                        SCHOOL_TYPE          TERM_TYPE
--------------------------------------------------------------

EXAMPLE:

AUBURN UNIVERSITY              123456789ABCD
AUBURN, ALABAMA  36530               S                QTR
```

Next the user will be able to select the exact function he wishes to perform with respect to the chosen school. Each update program works exactly like those for the student. School functions may be chosen using menu V3.2. Options C, D and E may only be visable to those users with sufficient access.

```
MENU V3.2:  SELECT SCHOOL FUNCTION

SCHOOL HEADER (D3)
SELECT THE DESIRED FUNCTION TO BE PERFORMED:  [ ]
     A:  UPDATE/EXAMINE INVOICES AND VOUCHERS
     B:  UPDATE/EXAMINE POINTS OF CONTACT
     C:  UPDATE/EXAMINE COST MATRIX
     D:  PRODUCE COST FORCASTS
     E:  PRODUCE CURRENT COSTS
     Q:  QUIT
```

If the user chooses option A from menu V3.2, he will be able to update and examine all the vouchers and invoices for the school along with all the detailed information relevant to them. Menu V3.2.1 will be used to display, in an update format, the current invoice data in the database for the school. The user will be able to add, change or delete information as required.

MENU V3.2.1: UPDATE/EXAMINE INVOICES AND VOUCHERS

```
SCHOOL HEADER (D3)
          INVOICE_NO   DATE_REC'D   INVOICE_AMOUNT  TERM_TYPE  TERM_DATES
          VOUCHER_NO      DATE_PAID      AMOUNT_PAID
LINE FUN
---------------------------------------------------------------------------
1   [ ] [         ]  [       ]   $[          ]     [  ] [         -        ]
         [         ]    [       ]   $[          ]
2   [ ] [         ]  [       ]   $[          ]     [  ] [         -        ]
         [         ]    [       ]   $[          ]
 :
 :
```

EXAMPLE:

```
SCHOOL HEADER (D3)
          INVOICE_NO  DATE_REC'D   INVOICE_AMOUNT  TERM_TYPE  TERM_DATES
          VOUCHER_NO     DATE_PAID      AMOUNT_PAID
LINE FUN
---------------------------------------------------------------------------
1   [ ] [12345678]   [23NOV79]    $[99999.99]     [SEM] [01DEC79-05JAN80]
         [ABCDEFGH]     [10DEC79]    $[55555.00]
```

If the user had chosen option B from menu V3.2 he would be able to examine and/or update the points of contact for each school and program. This is accomplished using menu V3.2.2 just as any other update display.

MENU V3.2.2: UPDATE POINTS OF CONTACT

```
SCHOOL HEADER (D3)
             NAME                                    PHONE          PROGRAM
                    ADDRESS
LINE FUN
---------------------------------------------------------------------------
1   [ ] [                                ]   [            ]  [           ]
         [                                                 ]
2   [ ] [                                ]   [            ]  [           ]
         [                                                              ]
 :
 :
```

EXAMPLE:

```
SCHOOL HEADER
             NAME                                        PHONE         PROGRAM
                     ADDRESS
LINE FUN
----------------------------------------------------------------------------------
 1  [ ] [DR. MICKEY M. MOUSE              ]       [999-123-456]  [COMP SCI      ]
        [COMPUTER SCIENCE DEPT, UNIV OF DISNEY WORLD, ORLANDO FLA               ]
```

Option C fom menu V3.2 the user to update the entries on the cost matrix for a particular school or simply generate it. The format and content of this function and its display have not been decided upon by the user and cannot be covered in detail here. What is known at this time is that the matrix will need to be able to be specified for a given year and program. It will contain a list of the charges the school levies against the Air Force under its contract along with the costs of all the various fees, tuition rates and any other cost related charges.

The user will be able to specify the year and program using menu V3.2.3, and in turn select the actual desired costs using menu V3.2.3.1.

MENU V3.2.3: UPDATE AND/OR GENERATE COST MATRIX

```
    SCHOOL HEADER (D3)
    ENTER THE DESIRED YEAR AND/OR PROGRAM
         YEAR [     ]
         PROGRAM [        ]
```

```
    MENU V3.2.3.1:  SELECT THE DESIRED COST

    SCHOOL HEADER
    ENTER THE DESIRED COST [ ]
         A:  TUITION
         B:  FEES
         Q:  QUIT
```

The remainder of the FSS has not been finalized by the user. Formats and content have yet to be decided. It is anticipated that the system would behave as described below.

The last two options will not have a menu associated with them because these functions simply generate a display or a printout. This information will have to be computed from other data in the database as needed and it is very likely that these functions could consume large amounts of time during production. Therefore, they should be produced off-line or in a batch mode. This will free the terminal for other work and provide the user the ability to view the report at a later time and at his leisure.

The associated displays should simply notify the user that he has requested this option and the reports are being generated. Once complete or a batch job has been spawned, control will return back to menu V3.2 and allow the user to perform other work.

MENU V3.2.4: PRODUCE COST FORCASTS

SCHOOL HEADER (D3)
PRODUCING COST FORCASTS FOR: (actual school name)

MENU V3.2.5: PRODUCE CURRENT COSTS

SCHOOL HEADER (D3)
PRODUCING CURRENT COSTS FOR: (actual school name) MENU V3.2.5: PRODUCE CURRENT COSTS

MODULE DESCRIPTIONS FOR THE AFIT/CIV FINANCIAL SUB SYSTEM (FSS) SOFTWARE

Further information on any of the menus or displays can be found in the Functional Description (FD) of the CFS software, Appendix IIIF.

MODULE 1

```
*****************************************************************  *
*                                                                  *
*NAME:  CIV FINANCIAL SUBSYSTEM                                     *
*                                                                  *
*FUNCTION:  This is the main program for the CFS and handles the   *
*           calls for the user.                                    *
*                                                                  *
*INPUT: USERID                                                     *
*OUTPUT:  NONE                                                     *
*****************************************************************  *
 PROCEDURE:
   do while  FUNCTION not = 'QUIT'
     call GET_FSS_FUNCTION (FUNCTION)
     if FUNCTION = 'STUDENT PAYMENTS'
       then call PROCESS_STUDENT (FUNCTION)
     else if FUNCTION = 'SCHOOL_PAYMENTS'
       then call PROCESS_SCHOOL (FUNCTION)
     else if FUNCTION = 'QUIT' then do nothing
  end main loop
```

MODULE 2

```
*****************************************************************  *
*                                                                  *
*NAME:   GET_FSS_FUNCTION                                          *
*                                                                  *
*FUNCTION:  This subroutine returns a single, valid function       *
*           which corresponds to which subroutine the user wishes  *
*           to enter.                                              *
*                                                                  *
*INPUT:  NONE                                                      *
*OUTPUT:  FUNCTION                                                 *
*****************************************************************  *
 PROCEDURE:
   error = 1b  ;a binary value turned on for loop control
   do while input not = A, B, Q  ;do while input is not valid
     print display lines on crt (display V1)
     error = 0b  ;turn it off until the user makes an error
     get input line
     if input not = Q then do
       if input = 'A' then FUNCTION = 'STUDENT PAYMENTS'
   else if input = 'B' then FUNCTION = 'SCHOOL PAYMENTS'
   else do
    error = 1b  ;turn error indicator on
    print error message for invalid function
   end
  end
  else FUNCTION = 'QUIT'
 end main loop
```

```
MODULE 3
*******************************************************************  *
*                                                                    *
*NAME:  PROCESS_STUDENT                                              *
*                                                                    *
*FUNCTION:  This subroutine acts as the driver for the Student      *
*           Financial Sub System (STFSS) and calls the functions    *
*           the user requests.                                      *
*                                                                    *
*INPUT:  NONE                                                        *
*OUTPUT:  FUNCTION (indicates a request to quit)                    *
*******************************************************************  *
 PROCEDURE:
  do while FUNCTION not = 'QUIT'
   call GET_STUDENT_SSAN (FUNCTION, SSAN)
   if FUNCTION not = 'QUIT' then do
    retrieve data for student header
    display data on CRT in the D1 format, as a protected fie
    call SELECT_STUDENT_FUNCTION (FUNCTION)
    if FUNCTION not = 'QUIT' then do
     if FUNCTION = 'INVOICES' then call UPDATE_INVOICES (FUNCTION, SSAN)
     else if FUNCTION = 'COSTS'
      then call QUERY_PROG_COSTS (FUNCTION, SSAN)
         else if FUNCTION = 'STUDENT' do nothing  ;get another ssan
    end
   end
  end main loop
```

MODULE 4

```
*******************************************************************  *
*                                                                    *
*NAME:  PROCESS_SCHOOL                                               *
*                                                                    *
*FUNCTION:  This subroutine acts as the driver for the School        *
*           Financial Sub System (SFSS) and calls the functions      *
*           the user requests.                                       *
*                                                                    *
*INPUT:  NONE                                                        *
*OUTPUT:  FUNCTION                                                   *
*******************************************************************  *
  PROCEDURE:
    do while FUNCTION not = 'QUIT'
      call GET_SCHOOL (FUNCTION, SCHOOL_NAME)
      if SCHOOL_NAME not = 'QUIT' then do
        retrieve school header data  ;screen display D3
        display school header data on crt
        call SELECT_SCHOOL_FUNCTION (FUNCTION)  ;menu V3.2
        if FUNCTION not = 'QUIT' then do
          if FUNCTION = 'INVOICES & VOUCHERS'
            then call UPDATE_INVOICES/VOUCHERS (FUNCTION, SCHOOL_NAME)
          else if FUNCTION = 'POC'
            then call UPDATE_POCS (FUNCTION, SCHOOL_NAME)
          else if FUNCTION = 'MATRIX'
            then call UPDATE_COST_MATRIX (FUNCTION, SCHOOL_NAME)
          else if FUNCTION = 'FORCAST'
            then call PRODUCE_COST_FORCASTS (FUNCTION, SCHOOL_NAME)
          else call CURRENT_COSTS (FUNCTION, SCHOOL_NAME)
        end if block
      else FUNCTION = 'QUIT'
    end main loop
```

MODULE 5

```
*******************************************************************  *
*                                                                    *
*NAME:  GET_STUDENT_SSAN                                             *
*                                                                    *
*FUNCTION:  This subroutine returns a unique SSAN for a student      *
*           using menu V2.  The user may quit by entering a          *
*           'Q' in SSAN.                                             *
*                                                                    *
*INPUT:  NONE                                                        *
*OUTPUT:  SSAN, FUNCTION                                             *
*******************************************************************  *
  PROCEDURE:
    error = 1b  ;a binary value turned on for loop control
    do while FUNCTION not = 'QUIT' & error
      error = 0b ;turn it off until the user makes an error
      display menu V2
      get input line
      check input for all numeric
      if input not numeric then do
        error = 1b
        print message for incorrect function
      if input = 'Q' then FUNCTION = 'QUIT'
      end
    end main loop
```

```
MODULE 6
****************************************************************. *
*                                                                 *
*NAME:  SELECT_STUDENT_FUNCTION                                   *
*                                                                 *
*FUNCTION:  This subroutine gets the users function from Menu V2..*
*           and returns it to the calling program.                *
*                                                                 *
*INPUT:  NONE                                                     *
*OUTPUT:  FUNCTION                                                *
****************************************************************. *
  PROCEDURE:
    error = 1b  ;a binary value turned on for loop control
    do while FUNCTION not = 'QUIT' & error
      error = 0b  ;turn it off until the user makes an error
      display menu V2.1
      get input line
      if input not = 'Q' then do
        if input = 'A' then FUNCTION = 'INVOICES'
        else if input = 'B' then FUNCTION = 'COSTS'
        else if input = 'C' then FUNCTION = 'STUDENT'
        else do
          error = 1b  ;flag the error
          print an error message to the user
        end
      end
      else FUNCTION = 'QUIT'
     end main loop
```

```
MODULE  7
****************************************************************** **
*                                                                   *
*NAME:  UPDATE_INVOICES                                             *
*                                                                   *
*FUNCTION:  This subroutine allows the user to update the invoice   *
*           charged to a particular student.  Each time a command   *
*           (function) is issued, the data is changed in the        *
*           database the data is retrieved again and the screen     *
*           is refreshed with the latest copy of the invoices as    *
*           stored in the database.  This subroutine uses Menu      *
*           V2.1.1 and display D3.                                  *
*                                                                   *
*INPUT:  STUDENT_SSAN                                               *
*OUTPUT:  FUNCTION                                                  *
****************************************************************** **
  PROCEDURE:
    do while FUNCTION not = 'QUIT & error
      error = 0b  ;error is a binary value, turn it off
      retrieve student invoices data
      display student invoice data in menu V2.1.1 format
      call GET_INVOICE_LINE (input line)
        ;ifun is the second field in the input line
        ;a check must be made to insure the line number was not changed
         by the user
      if ifun not = 'Q' & line_no not changed then do
        if ifun = '+' or '-' then move display lines up or down on screen
        else if ifun = 'A' the call ADD_INVOICE_TUPLE (data fields)
        else if ifun = 'C' then call CHANGE_INVOICE_TUPLE (data fields)
        else if ifun = 'D' the call DELETE_INVOICE_TUPLE (data fields)
        else do
          error = 1b  ;turn error on
          print error message for invalid function
        end
      else FUNCTION = 'QUIT'  ;quit this operation
     end main loop
```

NOTES: See the final section for an explanation of the A, C and D functions
an modules.

MODULE 8

```
*****************************************************************  *
*                                                                  *
*NAME:  GET_INVOICE_LINE                                           *
*                                                                  *
*FUNCTION:  This subroutine gets the data line from Menu V2.1.1    *
*           and checks the values of the fields for validity.      *
*           It will not return to the calling program until all    *
*           fields are correct or a 'Q' has been entered as        *
*           function.                                              *
*                                                                  *
*INPUT:  NONE                                                      *
*OUTPUT:  one  data line with fields separated by a delimiter      *
*****************************************************************  *
  PROCEDURE:
    error = 1b  ;a binary value turned on for loop control
    do while error
      error = 0b  ;turn it off until the user makes an error
      get line
      if ifun not = 'q' then do  ;ifun is the second field on the input line
        validate all field values for format and content
        query database to validate the AMOUNT_TYPE  ;see if it is there
        if any field fails then do
          error = 1b  ;flag the error
          print an appropriate error message
        end
    end main loop
```

MODULE 9

```
*****************************************************************  *
*                                                                  *
*NAME:  QUERY_PROG_COSTS                                           *
*                                                                  *
*FUNCTION:  This subroutine allows the user to query the various   *
*           costs incurred by a student in a particular program.   *
*           It uses menu V2.1.2 and display D2.                    *
*                                                                  *
*INPUT:  STUDENT_SSAN                                              *
*OUTPUT:  FUNCTION                                                 *
*****************************************************************  *
  PROCEDURE:
    do while FUNCTION not = 'QUIT'
      retrieve student cost data
      call GET_COST_TYPE (COST_TYPE, FUNCTION)
      if COST_TYPE = 'TUITION' then retrieve tuition data (for the SSAN)
      else if COST_TYPE = 'LAB' then retrieve lab fee data
      else if COST_TYPE = 'BOOKS' then retrieve book data
      else if FUNCTION not = 'QUIT' then display data in the format for D2
    end main loop
```

NOTES: This function could not be fully defined by the user at this time.

MODULE 10

```
**********************************************************************
*                                                                    *
*NAME:   GET_COST_TYPE                                               *
*                                                                    *
*FUNCTION:  This subroutine allows the user to select the type o     *
*           costs he wishes to update.                               *
*                                                                    *
*INPUT:  NONE                                                        *
*OUTPUT:  COST_TYPE, FUNCTION                                        *
**********************************************************************
  PROCEDURE:
    error = 1b  ;a binary value turned on for loop control
    do while error
      error = 0b  ;turn it off until the user makes an error
      display menu V2.1.2
      get line
      if input not = 'Q' then do
        if input = 'A' then COST_TYPE = 'TUITION'
        else if input = 'B' the COST_TYPE = 'LAB'
        else if input = 'C' then COST_TYPE = 'BOOKS'
      end
      else do
        error = 1b  ;flag the error for the loop
        print the appropriate error message
      end
        else if input = Q then FUNCTION = 'QUIT'
    end main loop
```

MODULE 11

```
**********************************************************************
*                                                                    *
*NAME:   GET_SCHOOL                                                  *
*                                                                    *
*FUNCTION:  This subroutine returns a single, valid name of the      *
*             school the user wishes to query/update.  If the nam    *
*           is unknown, the user may request a list of schools i     *
*           a particular state and choose one from the list.         *
*                                                                    *
*INPUT:  NONE                                                        *
*OUTPUT:    SCHOOL_NAME, FUNCTIONON                                    *
**********************************************************************
  PROCEDURE:
    error = 1b  ;a binary value turned on for loop control
    do while FUNCTION not = 'QUIT' & error
      error = 0b  ;turn it off until the user makes an error
      call GET_SCHOOL_NAME (SCHOOL_NAME, STATE, FUNCTION)
      check the school name in the data base
      if not valid or does not exist then do
        error = 1b  ;flag the error for the loop
        print an error message for an unknown school
      end
      else if FUNCTION not = 'QUIT' then
        if SCHOOL_NAME = blanks & STATE not = blanks
          then call GENERATE_SCHOOL_LIST (STATE, SCHOOL_NAME, FUNCTIONON)
        else if SCHOOL_NAME & STATE = blanks then do
          error = 1b  ;flag the error for the loop
          print an error message for invalid function
        end
    end main loop
```

```
MODULE 12
*******************************************************************  *
*                                                                   *
*NAME:   GET_SCHOOL_NAME                                            *
*                                                                   *
*FUNCTION:  This subroutine displays menu V3 and returns a singl    *
*           school name by allowing the user to enter it directl    *
*           or request a list of schools in a state.                *
*                                                                   *
*INPUT:  NONE                                                       *
*OUTPUT:   SCHOOL_NAME, FUNCTIONON                                    *
*******************************************************************  *
  PROCEDURE:
    error = 1b  ;a binary value turned on for loop control
    do while FUNCTION not = 'QUIT' & error
      error = 0b  ;turn it off until the user makes an error
      display menu V3
      get input
      if ifun = 'Q' then FUNCTION = 'QUIT'  ;ifun is the third field
      else do
        if SCHOOL_NAME = blanks & STATE not = blanks  ;user wants a list
          then validate state  ;uses the standard 2 letter abbrev
        if STATE is invalid then do
          error = 1b  ;flag the error for the loop
          print an error message for state not found
        end
    end main loop
```

MODULE 13

```
*******************************************************************  *
*                                                                    *
*NAME:    GENERATE_SCHOOL_LIST                                      *
*                                                                    *
*FUNCTION:  This subroutine takes a 2 letter state abbreviation     *
*           and retrieves all the AFIT/CI schools which are there, *
*           and allows the user to choose one from menu V3.1.       *
*                                                                    *
*INPUT:  STATE                                                       *
*OUTPUT:  SCHOOL_NAME, FUNCTION                                      *
*******************************************************************  *
  PROCEDURE:
    retrieve schools for the state
    display data in the format of V3.1
    error = 1b  ;a binary value turned on for loop control
    do while FUNCTION not = 'QUIT' & error
      error = 0b  ;turn it off until the user makes an error
      get input line
      check line number for correctness
      if too large or garbage then do
        error = 1b  ;flag the error for the loop
        print a nasty message to not change line numbers
      end
      else if line number points to quit (uses an array to store values)
        then FUNCTIONON = 'QUIT'
      else SCHOOL_NAME = data_array (line_no).school_name
    end main loop
```

MODULE 14

```
******************************************************************* **
*                                                                    *
*NAME: SELECT_SCHOOL_FUNCTION                                        *
*                                                                    *
*FUNCTION:  This subroutine allows the user to select the function*
*           he wishes to perform for a specific school from menu     *
*           V3.2.                                                    *
*                                                                    *
*INPUT:  NONE                                                        *
*OUTPUT:   FUNCTION                                                  *
******************************************************************* **
  PROCEDURE:
    error = 1b  ;a binary value turned on for loop control
    do while FUNCTION not = 'QUIT' & error
      error = 0b  ;turn it off until the user makes an error
      display menu V3.2
      get input
      if input not = 'Q' then do
        if input = 'A' then FUNCTION = 'INVOICES/VOUCHERS'
        else if input = 'B' then FUNCTION = 'POC'
        else if input = 'C' then FUNCTION = 'MATRIX'
        else if input = 'D' then FUNCTION = 'FORCAST'
        else if input = 'E' then FUNCTIONON = 'ONGOINC'
        else do
          error = 1b  ;flag the error for the loop
          print and error message for invalid input
        end
      else FUNCTION = 'QUIT'
    end main loop
```

```
MODULE 15
**************************************************************  *
*                                                               *
*NAME:  UPDATE_INVOICES/VOUCHERS                                *
*                                                               *
*FUNCTION:  This subroutine is the main driver and allows the us *
*           to update/view the invoices and vouchers which belon *
*           to a particular school.  It uses menu V3.2.1.       *
*                                                               *
*INPUT:  SCHOOL_NAME                                            *
*OUTPUT:   FUNCTION                                             *
**************************************************************  *
  PROCEDURE:
    error = 1b  ;a binary value turned on for loop control
    do while FUNCTION not = 'QUIT' & error
      error = 0b  ;turn it off until the user makes an error
      retrieve vouchers and invoices data for the school
      display the voucher and invoice data in the format for V 3. .1
      get input line
      ;ifun is the second field on the input line
      if ifun not = 'Q' then do
        check all the input fields
        check the database for the validity of the term_type
        if errors then do
          error = 1b  ;flag the error for the loop
          print appropriate error message
        end
        if ifun = '+' or '-' then move display lines up or down   screen
        else if ifun = 'A' then call ADD_INVOICE_TUPLE (input dat )
        else if ifun = 'C' then call CHANCE_INVOICE_TUPLE (input  ata)
        else if ifun = 'D' then call DELETE_INVOICE_TUPLE (input  ata)
      end
      else if line numbers changed then do
        error = 1b  ;flag the error for the loop
        print nasty message not to change the line numbers
      end
      else if ifun = 'Q' then FUNCTION = 'QUIT'
     end main loop

NOTES:  See the final section for an explanation of the A, C and   functions
```

and modules.

```
MODULE  16
*****************************************************************   *
*                                                                   *
*NAME:    UPDATE_POCS                                              *
*                                                                   *
*FUNCTION:  This subroutine is the driver for updating the          *
*           Points of Contact for a given school.  It uses menu     *
*           V3.2.2.                                                 *
*                                                                   *
*INPUT:  SCHOOL_NAME                                                *
*OUTPUT:   FUNCTION                                                 *
*****************************************************************   *
  PROCEDURE:
    error = 1b  ;a binary value turned on for loop control
    do while FUNCTION not = 'QUIT' & error
      error = 0b  ;turn it off until the user makes an error
      retrieve POC data for the school
      display POC data in the format of V3.2.2
      get input line
      ;  ifun is the second field on the input line
      if ifun not = 'Q' & line number not changed then do
        check all input fields for validity
        if errors then do
          error = 1b  ;flag the error for the loop
          print the appropriate error message
        end
        else do
          if ifun = '+' or '-' then move display lines up or dow  on screen
          else if ifun = 'A' then call ADD_POC_TUPLE (input data)
          else if ifun = 'D' then call DELETE_POC_TUPLE (input data)
          else if ifun = 'C' then call CHANGE_POC_TUPLE (input data)
          else do
            error = 1b  ;flag the error for the loop
            print error message for invalid function
          end
      else if line number changed then do
        error = 1b  ;flag the error for the loop
        print nasty message to not change line numbers
      end
      else if ifun = 'Q' then FUNCTION = 'QUIT'
    end main loop

NOTES:  See the final section for an explanation of the A, C and   functio
        and modules.
```

```
MODULE 17
********************************************************************
*                                                                  *
*NAME:   UPDATE_COST_MATRIX                                        *
*                                                                  *
*FUNCTION:  This subroutine acts as the main driver which will     *
*           allow the user to update the cost matrix using menu    *
*           V3.2.3.1.                                              *
*                                                                  *
*INPUT:  SCHOOL_NAME                                               *
*OUTPUT:   FUNCTION                                                *
********************************************************************
  PROCEDURE:
    error = 1b  ;a binary value turned on for loop control
    do while FUNCTION not = 'QUIT' & error
      error = 0b  ;turn it off until the user makes an error
      call GET_YEAR_AND_PROGRAM (YEAR, PROGRAM, FUNCTION)
      display menu V 3.2.3.1
      get input
      check input for validity
      if not valid then do
        error = 1b  ;flag the error for the loop
        print and error message for invalid function
      end
      if not error & input not = 'Q' then do
        if input = 'A' then
          call UPDATE_TUITION_MATRIX (SCHOOL_NAME, YEAR, PROGRAM, FUNCTION)
        else call UPDATE_FEE_MATRIX (SCHOOL_NAME, YEAR, PROGRAM, FUNCTION)
      end
    end main loop
```

```
MODULE 18
********************************************************************
*                                                                  *
*NAME:   FORCAST_COSTS                                             *
*                                                                  *
*FUNCTION:  This subroutine allows the user to update the cost fore-*
*           casting information.  It is the main driver for this   *
*           function and uses menu V3.2.4.                         *
*                                                                  *
*INPUT:                                                            *
*OUTPUT:                                                           *
********************************************************************
  PROCEDURE:

THERE IS INSUFFICIENT DATA AT THIS TIME TO DESCRIBE THE PROCEDURE OF THIS
  MODULE.
```

MODULE 19

```
*******************************************************************  *
*                                                                    *
*NAME:   CURRENT_COSTS                                               *
*                                                                    *
*FUNCTION:  This subroutine allows the user to update or examine     *
*           the current costs incurred for a given school.  It       *
*           uses menu V3.2.5.                                        *
*                                                                    *
*INPUT:                                                              *
*OUTPUT:                                                             *
*******************************************************************  *
  PROCEDURE:
```

THERE IS INSUFFICIENT DATA AT THIS TIME TO DESCRIBE THE PROCEDURE OF THIS MODULE.

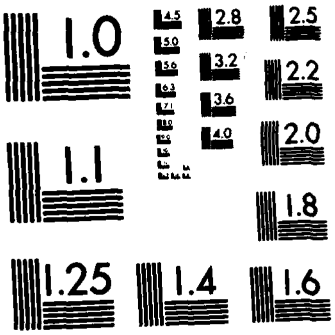
MODULE 20

```
*******************************************************************  *
*                                                                    *
*NAME:   GET_YEAR_AND_PROGRAM                                        *
*                                                                    *
*FUNCTION:  This subroutine gets a year and/or a program using       *
*           menu 3.2.2.                                              *
*                                                                    *
*INPUT:  NONE                                                        *
*OUTPUT:   YEAR, PROGRAM, FUNCTION                                   *
*******************************************************************  *
  PROCEDURE:
    error = 1b  ;a binary value turned on for loop control
    do while input not = 'Q' & error
      error = 0b  ;turn it off until the user makes an error
      display menu 3.2.2
      get line
      check input fields
      if errors then do
        error = 1b  ;flag the error for the loop
        print an error message for invalid input
      end
    end main loop
```

AD-A124 647 ANALYSIS OF INFORMATION REQUIREMENTS AND DESIGN OF THE CONSOLIDATED AFIT D..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. 4/4

UNCLASSIFIED J S RICKS ET AL. DEC 82 AFIT/GCS/EE/82D-11 F/G 5/2 NL

END
FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

MODULE 21

```
******************************************************************* *
*                                                                   *
*NAME:   UPDATE_TUITION_MATRIX                                      *
*                                                                   *
*FUNCTION:  This subroutine acts as the driver which will allow     *
*           the user to update/examine the tuition cost matrix.     *
*                                                                   *
*INPUT:  YEAR, SCHOOL_NAME, PROGRAM                                 *
*OUTPUT:  FUNCTION                                                  *
******************************************************************* *
  PROCEDURE:
    error = 1b  ;error is a binary value, set if for the loop
    do while ifun not = 'Q' & error
      error = 0b  ;turn it off
      retrieve tuition matrix data
      display tuition matrix  ;format has not been finalized by the user
      get input line
      ;ifun is the second field on the input line
      ; line numbers must be checked
      if ifun not = 'Q' & line numbers not changed then do
        check all inputs
        if errors then do
          error = 1b  ;flag the error for the loop
          print the appropriate error message
         end
        if ifun = '+' or '-' then move display lines up or down on screen
        else if ifun = 'A' then call ADD_TUITION_TUPLE (input data)
        else if ifun = 'C' then call CHANGE_TUITION_TUPLE (input data)
        else if ifun = 'D' then call DELETE_TUITION_TUPLE (input data)
      end
    end main loop
```

NOTES: See the final section for an explanation of the A, C and D functions and modules.

MODULE 22

```
******************************************************************* *
*                                                                   *
*NAME:   UPDATE_FEE_MATRIX                                          *
*                                                                   *
*FUNCTION:  This subroutine is the driver which allows the user to  *
*           update/view the various fees charged by a school.       *
*                                                                   *
*INPUT:  YEAR, PROGRAM, SCHOOL_NAME                                 *
*OUTPUT:   FUNCTION                                                 *
******************************************************************* *
  PROCEDURE:
    error = 1b  ;error is a binary value, set it for the loop
    do while ifun not = 'Q' & error  ;ifun is the second field on the input line
    error = 0b  ;turn it off
    retrieve fee matrix data
    display fee matrix  ;format is not yet finalized by the us
    get input
    check all input for validity
    if errors then do
      error = 1b  ;flag the error for the loop
      print appropriate error message
    end
    if ifun not = 'Q' & line numbers not changed then do
      if ifun = '+' or '-' then move display lines up or down . . screen
      else if ifun = 'A' then call ADD_FEE_TUPLE (input data)
      else if ifun = 'C' then call CHANGE_FEE_TUPLE (input dat
      else if ifun = 'D' then call DELETE_FEE_TUPLE (input dat
      else if ifun = 'Q' then FUNCTION = 'QUIT'
    end
  end main loop
```

NOTES: See the final section for an explanation of the A, C and   functions and modules.

DESCRIPTIONS OF THE
ADD, CHANGE AND DELETE MODULES

in +5

For of these modules, found in the update routines, each has the same basic procedure and arguments. The only difference between them is the actual database selection expressions, required to accurately identify the tuples (data) in question.

The 'ADD' modules will only take the data input by the user and add it to a relation, or relations depending upon what it was written for. Likewise the "DELETE' modules will only remove data specified by the user. In order to keep things simple, straightforward and help the readability of the code, a 'CHANGE' module will call the 'DELETE' then the 'ADD' modules respectively.

The above method of constructing the 'CHANGE' module is also preferred because most DBMSs will not allow a key field to be modified using a "change" or "modify" command. Therefore the only way to modify a key field is to first delete it then add the new one. Depending upon the actual DBMS implemented at AFIT, this may or may not have to be done this way.

NEW AND MODIFIED RELATIONS, NEW ATTRIBUTES, AND FUNCTIONAL DEPENDENCIES FOR AFIT/CIR/M

NEW RELATIONS

1. CI_SUSPENSE_TYPE:
   Type*, Description
   (See note 1)
   (Describes the various types of CI suspenses.)

2. CI_SUSPENSE:
   SSAN*, S_Date*, Type*, Description*
   (Contains data about a given suspense for a particular student.)

3. CI_TERM_TYPE:
   Term*, Description
   (Describes the various types of civilian academic institution terms.)

4. CI_TERM:
   SSAN*, Term*, Start_Date*, Stop_Date
   (Contains data about a particular term for a given student.)

5. CI_ED_PLAN:
   SSAN*, Course_ID, Course_Dept, Course_No*, Course_Title,
   Term_No*, Credits, Start_Date*
   (Contains data about a particular CI student's Ed Plan.)

6. CI_GRADES:
   SSAN*, Course_No*, Term_No*, Grade, Start_Date*
   (Contains data about the grade a particular CI student got for a given course.)

7. CI_Student_Manager:
   SSAN*, PM
   (Defines who a given CI student's program manager (PM) is.)

MODIFIED RELATIONS

1. CI_DATA:
   Add Advisor, Grad_Date, GRE_Verbal, GRE_Quant as attributes
   (See note 2)

NEW ATTRIBUTE DESCRIPTIONS

ATTRIBUTE	RELATIONS	DESCRIPTION	EXAMPLE	SIZE
xxxxxxxxx	xxxxxxxxx	xxxxxxxxxxx	xxxxxxx	xxxx
1. Course_ Dept	5 (See note 3)	Dept. offering a course	EE	2
2. Course_ ID	5	Course identifier	? (See note 4)	TBD
3. Course_ No	5*,6* (See note 5)	Course number	7.93	4
4. Course_ Title	5	---	DATABASE 1	30
5. Start_ Date	4*,5*,6*	Date a term started or will start	090182	6
6. PM	7	Program manager Also is the last 2 char of the CI users ID	RQ	2
7. Stop_ Date	4	Date a term stopped or will stop	121582	6
8. S_Date	2*	Suspense Date	121782	6
9. Type	1*,2*	Code for type of suspense	?	TBD
10. Term	3*,4*	Code for type	S1 (Summer 1 Q (Quarter)	2
11. Term_No	5*,6*	Number of a particular term	1	1

FUNCTIONAL DEPENDENCIES

RELATION	FUNCTION DEPENDENCIES	IMPORTANT AS MPTIONS
xxxxxxxx	xxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxx xxxxxxx
1	1->2, 2->1 (See notes 6 & 7)	Type is uniq
2	1234->1234	---
3	1->2, 2->1	Term is uniq
4	123->4	---
5	1468->2357	Course_No is nique for a given udent at a particu r school Course_Titl is not necessarily ique
6	1235->4 (See note 8)	---
7	1->2	---

NOTES

1. A starred attribute indicates that this attribute is a key o part of a key for the relation that it is starred in. 2. CI_Data s considered a modified relation because it was defined prior to the in-depth analysis of AFIT/CI's needs. Also, the modi ied CI_ Data is still in Third Normal Form. 3. A number in the LATIONS column refers to the relation numbers in the NEW RELATIONS section. 4. A "?" and a "TBD" means that nsufficient data was available at the at the time this appendix was written and that this informat n needs to be developed. 5. A starred relation number has the same meaning as note 1. 6. X->Y means the set of attributes X functional y determines the set of attributes Y, e.g. 1->2 means attribute 1 functionally de rmines attribute 2. 7. Attribute numbers used in describing f ctional dependencies are based on the ordinal position of the attribute within the given re tion in the NEW RELATIONS section. For example, 1->2 means that the irst attribute in the given relation functionally determines the cond attribute. 8. All new relations are in Third Normal Form, sed on these FD's, and available information.

NEW OR MODIFIED RELATIONS FOR AFIT/CIV

1. CIV_FEES:
   (FEE_CODE*, TYPE, DESCRIPTION, EEIC_CODE)

   This relation contains a list of all of the various fees for which AFIT can be charged. FEE_CODE is a unique code and TYPE is a classification or category to which the fee belongs (e.g. several specific medical fees could all be classified as medfees).

2. INST_PAYMENTS:
   (LOCATION_CODE*, INVOICE_NO*, DATE_REC*, INVOICE_AMT, PMT_CODE*, VOUCHER_AMT, VOUCHER_NO, TERM_ST_DATE, TERM_END_DATE)

   This relation is the place where all the school transactions are kept. PMT_CODE is the same as FEE_CODE and TUITION_TYPE because a payment may be made for several kinds of costs.

3. INST_COSTS:
   (LOCATION_CODE*, TUITION_TYPE*, TUITION_RATE)

   This relation is a list of the various tuition rates which a school may charge. TUITION_TYPE is a code and a full description is in relation #5 below. Examples of these types are: undergrad, grad and medical.

4. INST_CHARGES:
   (LOCATION_CODE*, PROGRAM*, RC/CCC, FEE_CODE*, CHARGE)

   This relation is a list, by program, of the charges a school will/may levy against the Air Force for AFIT. This essentially is a rate schedule.

5. TUITIONS:
   (TUITION_TYPE*, DESCRIPTION)

   This relation is a list of the TUITION_TYPES and their descriptions.

6. SCHOOL/EWI_DATA:
   (LOCATION_CODE*, SERVICING_AFO, ESA_NO, SCHOOL_TYPE, STATE, TERM_TYPE)

   The contents and purpose of this relation remain unchanged. Two fields, ESA_NO and STATE have been added.

7. CI_DATA: (MODIFIED)
(....PROGRAM_MGR, FUNDING_TYPE)

The above two attributes should be added to the CI_DATA relation already defined.

8. FUNDING_TYPES:
(FUNDING_TYPE*, DESCRIPTION)

This relation is a list of the various funding types or sources for AFIT sponsored education. Examples are: funded legal,...

9. CI_STUDENT_PAYMENTS:
(SSAN*, START_DATE, PMT_CODE*, VOUCHER_AMT, AMT_REQUESTED, DATE_POSTED, END_DATE, VOUCHER_NO*, REMARKS)

This relation is the place where all the payments made to a student are kept. The vouchers here should be cross referenced by VOUCHER_NO to the INST_PAYMENTS relation. The attribute PMT_CODE is the same as FEE_CODE and TUITION_TYPE and is used the same as in relation #2.

10. ACAD_TERMS:
(TERM_TYPE*, DESCRIPTION)

This relation is a list of the various type of terms which AFIT/CIV must deal with (e.g. qtr, sem, summer sessions and trimesters)

11. SCHOOL_TYPES:
(SCHOOL_TYPE*, DESCRIPTION)

This relation is a list of the various types of schools which AFIT/CIV must deal with (e.g. state, private).

FUNCTIONAL DEPENDENCIES:

Based upon available data and knowledge, the above relations are in third normal form. Each set of key attributes (ones which are marked with an '*') wholly determines (i.e. no subset) all of the other attributes.

ATTRIBUTE DESCRIPTIONS:

Insufficient data was available to provide a detailed list of attribute descriptions. Several meetings were held with AFIT/CI personnel to clearly identify the remaining data needs, but the amount was much too large and some very important elements were not clear or used consistently enough to make accurate determinations in the available time. AFIT/CIV must make some operational decisions to standardize their data and its use before much of it will be able to be effectively automated.

VITAS

Capt Jeffrey S. Ricks was born at the Pensacola Naval Air Station, Pensacola Florida on 8 Dec 1953. He graduated from Foley High School, Foley Alabama in 1972 and is a 1976 AFROTC graduate of Auburn University, Auburn Alabama where he received a Bachelor of Science Degree in Secondary Mathematics Education. Upon receiving his commission in June 1976 he was assigned to the Air Force Data Services Center, the Pentagon where he served as a Headquarters Air Force Budget Systems Analyst. While there he was involved in the design, construction and implementation of a relational database centered, Management Information System (on the MULTICS computer system) for a portion of the Air Staff.

In 1979 Capt Ricks was reassigned to the Air Force Directorate of Computer Resources (AF/ACD) where he was an action officer responsible for the Automated Data Processing (ADP) requirements of the Pacific Air Forces (PACAF), Electronic Security Command (ESC), Tactical Air Command (TAC) and certain intelligence, World Wide Military Command and Control System (WWMCCS) and other special, ADP related programs. In May 1981 he entered the Air Force Institute of Technology, School of Engineering where he pursued a Masters Degree in Computer Systems.

Permanent Address: 40 Village Green, Magnolia Springs, Alabama 36536

VITAS

Lt Robert Steven Colburn was born in Ransom, Kansas on 8 Oct 1953. He graduated from Mannford Senior High School, Mannford, Oklahoma in 1971. After graduation he joined the U.S. Navy, where he accumulated over 2000 hours flight time as a Second Mechanic (Assistant Flight Engineer), including missions in Viet Nam. Upon completion of his Navy tour in 1975, he entered Garden City Community College, Garden City, Kansas where he received an Associates Degree. He continued his education at the University of Kansas, Lawrence, Kansas, receiving a Bachelor of Arts Degree in Computer Science and an AFROTC commission in 1981.

In May 1981 Lt Colburn entered the Air Force Institute of Technology, School of Engineering where he pursued a Masters Degree in Computer Systems with an emphasis on Information Systems.

Permanent address: 1605 Kansas Ave.
Atchison, Ks. 66002